특2002-0029127

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

Conn. WO 01/19028 A2

(51) Int. Cl.
H04L 12/28

(11) 공개번호 특2002-0029127
(43) 공개일자 2002년04월17일

(21) 출원번호 10-2002-7003168
(22) 출원일자 2002년03월09일
번역문제출일자 2002년03월09일
(86) 국제출원번호 PCT/US2000/40835 (87) 국제공개번호 WO 2001/19028
(86) 국제출원출원일자 2000년09월06일 (87) 국제공개일자 2001년03월15일
(81) 지정국 국내특허 : 알바니아 아르메니아 오스트리아 오스트레일리아 아제르바이잔 보스니아-헤르체고비나 바베이도스 불가리아 브라질 벨라루스 캐나다 스위스 중국 쿠바 체코 독일 덴마크 에스토니아 스페인 핀란드 영국 그루지야 헝가리 이스라엘 아이슬란드 일본 케냐 키르기즈 북한 대한민국 카자흐스탄 세인트루시아 스리랑카 라이베리아 레소토 리투아니아 룩셈부르크 라트비아 몰도바 마다가스카르 마케도니아 몽고 말라위 멕시코 노르웨이 뉴질랜드 슬로베니아 슬로바키아 타지키스탄 투르크메니스탄 터어키 트리니다드토바고 우크라이나 우간다 우즈베키스탄 베트남 폴란드 포르투칼 루마니아 러시아 수단 스웨덴 싱가포르 아랍에미리트 안티구아바부다 코스타리카 도미니카연방 알제리 모로코 탄자니아 남아프리카 벨리즈 모잠비크 그레나다 가나 감비아 크로아티아 인도네시아 인도 시에라리온 유고슬라비아 짐바브웨
AP ARIPO특허 : 케냐 레소토 말라위 수단 스와질랜드 우간다 시에라리온 가나 감비아 짐바브웨 모잠비크 탄자니아
EA 유라시아특허 : 아르메니아 아제르바이잔 벨라루스 키르기즈 카자흐스탄 몰도바 러시아 타지키스탄 투르크메니스탄
EP 유럽특허 : 오스트리아 벨기에 스위스 독일 덴마크 스페인 프랑스 영국 그리스 아일랜드 이탈리아 룩셈부르크 모나코 네덜란드 포르투칼 스웨덴 핀랜드 사이프러스
OA OAPI특허 : 부르키나파소 베넹 중앙아프리카 콩고 코트디브와르 카메룬 가봉 기네 말리 모리타니 니제르 세네갈 차드 토고 기네비쏘

(30) 우선권주장 60/153,148 1999년09월09일 미국(US)
09/588,619 2000년06월06일 미국(US)
(71) 출원인 에비시 시스템스 추후보정
미국 매사추세츠 노쓰 빌리카 빌리카 로드 빌딩6 101
(72) 발명자 델리,힐리앙,제이.
미국94305캘리포니아스탠포드베르니어플레이스1068
카베이,필립,피.
미국01730매사추세츠베드포드다니엘스로드7
벨리보,폴,에이.
미국02474매사추세츠알링튼게리슨로드5
만,윌리암,에프.
미국01776매사추세츠서드베리체커베리서클23
데니슨,레리,알.
미국02062매사추세츠노르우드나단스트리트505
(74) 대리인 남상선

심사청구 : 없음

(54) 패킷 스케쥴링 방법 및 장치

요약

네트워크 라우터에서, 트리 구조 또는 분류화 네트워크는 스케쥴링 값을 비교하며 적당한 큐로부터 포워딩된 패킷을 선택하는데 이용된다. 트리구조에서, 각각의 잎(leaf)은 큐의 스케쥴링값을 나타내며, 이 구조의 내부 노드는 트리 구조의 형제 노드의 스케쥴링값의 비교시 승자를 나타낸다. CBR 스케쥴링값은

40-1

큐를 선택하기위하여 가장 먼저 비교되며, 만일 CBR로부터의 전송이 적절하지 않다면, 패킷은 WFQ 스케쥴링값을 사용하여 선택될 것이다. 스케쥴링값은 가변 패킷 길이 및 이전 패킷에 채워진 바이트를 반영하도록 업데이트된다. 스케쥴링은 여러 단계에서 수행될 수 있다.

도면의 간단한 설명

도1

발명의 상세한 설명

기술분야

본 발명은 패킷 스케쥴링 방법 및 장치에 관한 것이다.

배경기술

네트워크 상에서 경험할 수 있는 QoS(Quality of Service)는 딜레이, 지터 및 손실로 특성화된다. 많은 네트워크 애플리케이션들은 이러한 하나 이상의 파라미터들에 민감하다. 예컨대, 대화형 애플리케이션은 네트워크의 전체 딜레이가 임계값을 초과하면 사용할 수 없게 된다. 비디오와 같은 스트리밍 애플리케이션들은 대량의 고정 딜레이는 용인할 수 있지만 서로 다른 패킷들이 경험하는 딜레이의 지터 또는 변화가 임계값을 초과하면 초과의 버퍼 공간을 필요로 한다. 다른 애플리케이션들은 패킷 손실, 예컨대 초과 정체(congestion) 때문에 드롭되는 패킷들을 용인할 수 없다.

오늘날의 대부분의 인터넷 트래픽은 최선의 노력으로(best-effort basis) 전달되는 단일 부류의 트래픽으로서 조절된다. 즉, 네트워크는 모든 패킷들을 적정한 시간 내에 전달하려고 노력하지만 딜레이나 손실에 대해서는 보장이 없다. 네트워크 트래픽의 버스트 특성 때문에, 몇몇 패킷들은 정체 때문에 딜레이 또는 드롭될 수 있으며, 패킷들이 경험하는 딜레이는 패킷에 따라 상당히 다르게 된다. 최선의 노력의 전달은 파일-전달 및 웹 페이지 액세스와 같은 많은 타입의 트래픽에 충분하지만, 실시간 오디오 및 비디오 스트림과 같은 다른 타입의 트래픽에는 적절치 않다.

실시간으로 트래픽을 지원하고 또한 특정 고객에게 프리미엄 급의 서비스를 제공하기 위해, 몇몇 네트워크는 트래픽을 분류들로 분리하고 각 트래픽 부류에 대한 서비스 보장을 제공한다. 예를 들어, 트래픽은 제한 비트율 및 제한 딜레이가 보장되는 한 부류, 최소 비트율이 보장되는 제2 부류, 및 전체 네트워크 대역폭의 서로 다른 부분들이 할당되는 하나 이상의 최선 노력의 트래픽 부류들로 분할된다.

네트워크의 전달 유닛(패킷, 셀 또는 프레임)이 수신하는 서비스 부류는 많은 방식으로 유닛에 인코딩된다. 인터넷 패킷에서, 예컨대 서비스 부류는 패킷 헤더의 8비트 ToS(Type of Service) 필드를 사용하여 결정된다. 선택적으로, 서비스 부류는 패킷이 속하는 흐름으로 부터 유도된다. ATM 네트워크에서, 서비스 부류는 셀이 통과하는 가상 회로와 관련된다.

서비스 보장은 입력 정리(policing) 및 출력 스케쥴링의 조합에 의해 구현된다. 입력 정리는 라우터에 도달하는 모든 트래픽이 적절한 서비스 접속과 일치하도록 보장한다. 패킷이 라우터에 도달하면, 그 도달 시간이 그 서비스 접속과 일치하는지 결정하기 위한 체크가 행해진다. 패킷이 일치하면, 정상적으로 처리된다. 패킷이 일치하지 않으면, 예컨대 10Mbit/s의 대역폭이 보장된 흐름이 15Mbits/s를 소모하면, 표시가 된다. 패킷은 정상적으로 처리되고, 네트워크 정책에 따라 서비스가 품질저하 되거나 드롭된다.

출력 스케쥴링은 패킷이 특정 출력 채널을 통해 라우터를 떠나는 순서를 결정한다. 스케쥴링은 특정 QoS를 보장하는에 있어서 주요 팩터이다. 예를 들어, 제한 비트율을 특정 흐름으로 보장하기 위해, 이러한 흐름의 패킷들은 사용가능한 용량을 초과할 수 있는 버스트한 최선 노력의 흐름으로부터의 패킷의 앞에 스케쥴되어야 한다. 스트리밍 흐름 상에 낮은 지터를 보장하기 위해, 각 패킷은 좁은 시간의 윈도우 내에서 라우터를 떠나도록 스케쥴된다.

GBR(garanteed bit rate) 트래픽을 위한 출력 스케쥴링은 종종 도6에 설명된 다중 큐 구조를 사용하여 수행된다. 각 트래픽 부류(클래스 기반 큐잉(CBQ)) 또는 각 네트워크 흐름(흐름 큐잉당)에 대해 별개의 출력 큐(501-503)이 제공된다. 도면에는 세개의 큐들이 제시되어 있지만 다른 수의 큐들도 가능하다. 각 큐는 출력 라인(531)을 통한 전송을 기다리는 몇몇의 패킷들(예컨대, 패킷(521))을 기다린다. 다음 패킷이 큐를 떠날 때를 나타내는 카운터(511-513)는 각 큐와 관련된다. 예컨대, 일정 비트율 트래픽과 관련된 큐들에 대해서, 카운터는 다음 패킷이 큐를 떠나야 할 때를 표시한다. 패킷이 큐를 떠난 후에, 카운터는 현재 시간 및 대역폭 할당에 의해 분리된 패킷의 길이를 반영하는 증분으로 업데이트 된다. 이러한 업데이트 방법은 일정 비트율 QoS를 위한 ATM '누수-버킷(leaky-bucket) 알고리즘에 대응한다.

최선 노력의 트래픽을 위한 출력 스케쥴링은 또한 WFQ(weighted-fair queuing)를 사용하여 수행된다. 그러한 트래픽은 또한 각 트래픽 부류 또는 각 흐름에 대한 별개의 큐로 도6에 설명된 장치를 사용하여 스케쥴된다. 그러나, 최선 노력의 트래픽을 위해서는, 큐의 패킷들에는 출력 포트 상의 보장된 비트율이 할당되지 않았다. 오히려, 각각의 큐는 이용가능한 대역폭의 일부를 할당하며, 패킷은 이용가능한 대역폭에 따른 최상의 작용을 기반으로 하여 스케쥴링된다. 이러한 최상의 작용 트래픽과 연관된 큐에서, 각각의 큐와 연관된 카운터(511-513)는 가중된 페어 큐잉 알고리즘에 따라 실행되며, 출력 대역폭이 이용가능할때 가장 작은 카운트와 연관된 큐가 전송을 위하여 선택되며 그것의 카운터는 대역폭 '셰어(share)'의 역수만큼 증가된다.

종래의 라우터에서, 가장 낮은 카운터를 탐색하고, 적정 큐를 선택하며, 상기 큐와 연관된 카운터를 업데이트하는 과정은 마이크로프로세서상에서 실행되는 소프트웨어에 의하여 수행된다. 클래스 기반 큐잉 또는 퍼-플로우(per-flow) 큐잉으로 각각의 패킷을 전송하는데 필요한 처리 오버헤드때문에, 이들 메커니즘은 중간 갯수의 큐(10s 내지 100s)를 가지고 저속링크상에서 사용하도록 제한된다(Ferguson and Huston,

40-2

Quality of Service, Wiley, 1998, p. 61 참조)

발명의 상세한 설명

본 발명의 일 특징에 따르면, 네트워크 라우터는 전송될 데이터 패킷을 저장하는 큐를 포함한다. 패킷이 전송되는 큐를 선택하는 스케줄러는 GBR 및 WFQ 카운터 값과 같이 큐와 연관된 스케줄링 값을 유지한다. 스케줄링 값은 전송될 패킷을 선택하기 위하여 선택 네트워크에서 비교된다.

선택 네트워크는 트리 구조의 각 잎이 큐의 스케줄링 값을 나타내는 트리 구조일 수 있다. 트리 구조의 내부 노드는 트리 구조의 형제 노드에 대한 스케줄링 값과 비교하여 승자를 나타낸다. 선택적으로, 선택 네트워크는 스케줄링이 스케줄링 우선순위에 의하여 큐의 순서를 정하기 위하여 비교되는 저장 네트워크일 수 있다.

트리 구조에서, 스케줄러는 트리 구조의 뿌리에 대해 변화된 스케줄링 값을 나타내는 잎 노드로부터 트리 구조를 통한 경로에 대한 스케줄링 값의 비교를 제한한다.

트리 구조의 내부 노드는 승리한 형제 노드로부터 스케줄링 값을 저장할 수 있다. 내부 노드는 저장된 스케줄링 값에 대응하는 잎 노드의 식별자를 저장할 수 있다.

대안적인 트리 구조에서, 각각의 노드는 승리한 잎 노드에 대한 경로를 식별한다. 랜덤 액세스 메모리는 잎 노드만을 저장하는 반면에, 플립-플롭 어레이는 각각의 내부 노드에서 승자를 식별한다. 비교기는 플립-플롭 어레이에 의해 지시된 RAM에서 잎 노드의 스케줄링 값을 비교한다.

본 발명의 다른 특징에 따르면, 추가 표시기는 스케줄링으로부터 노드의 큐를 디스에이블하기 위하여 각 잎 노드와 연관될 수 있다.

스케줄러는 트리 구조를 저장하기 위한 랜덤 액세스 메모리를 포함할 수 있다. 어드레스 레지스터는 비교될 스케줄링 값을 RAM으로부터 액세스하기 위하여 어드레스를 저장한다. 비교 레지스터는 RAM으로부터의 스케줄링 값에 비교될 스케줄링 값을 저장하며, 비교기는 스케줄링 값을 비교한다. 스케줄러는 어드레스 레지스터에서 어드레스를 수신하는 하드웨어를 더 포함할 수 있으며, 비교될 스케줄링 값이 저장되는 형제 노드를 결정한다. 하드웨어는 승리한 비교 스케줄링 값이 저장되는 부모 노드 어드레스를 결정한다.

스케줄러는 파이프라인 단계를 포함할 수 있으며, 그 각각은 트리 구조의 분리된 부분에 의하여 지시된 스케줄링값을 비교한다. 랜덤 액세스 메모리는 파이프라인 단계를 따라 분할되며, 각각의 파티션은 트리 구조의 최하위 레벨에 저장한다. 각각의 파이프라인 단계는 어드레스 레지스터, 비교 레지스터, 및 비교기를 포함한다.

스케줄러는 제 1 큐 부세트와 연관된 제 1 스케줄링 방법에 대응하는 스케줄링 값과 제 2 큐 부세트와 연관된 제 2 스케줄링 방법에 대응하는 스케줄링 값을 포함할 수 있으며, 적어도 하나의 큐는 제 1 큐 부세트 및 제 2큐 부세트의 각의 그룹이다.

스케줄링 값은 일정 비트율(CBR) 서비스 계런티에 따라 스케줄링된 전송 시간을 포함할 수 있다. 스케줄링 값은 가중된 페어 큐잉(WFQ) 스케줄링 방법을 사용하여 이론적인 전송 시간을 부가적으로 나타낼 수 있다. 초기에 스케줄링된 CBR 큐가 우선 식별된다. 만일 식별된 CBR 큐의 스케줄링 값이 현재의 시간보다 늦거나 동일하다면, 대응하는 패킷이 CBR 큐로부터 전송되며 큐와 연관된 CBR 스케줄링 값이 업데이트된다. 그렇지 않으면, 패킷은 최초의 스케줄링 값을 갖는 WFQ 큐(queue)로부터 전송되고 상기 스케줄링 값은 업데이팅된다.

본 발명의 다른 측면에 따라, 스케줄링 값은 가변 패킷 길이를 반영하고 이전 패킷에 인가된 바이트 스터핑을 반영하도록 업데이팅된다.

본 발명의 또 다른 측면에 따라, 스케줄링은 다중 스테이지에서 수행된다. 데이터 패킷은 제 1 및 제 2 큐 세트에 저장된다. 제 1 스케줄러는 어느 패킷들이 제 1 중간 큐에 전송되는지로부터 큐들 중 제 1 세트의 큐들을 선택하고, 제 2 스케줄러는 어느 패킷들이 제 2 중간 큐에 전송되는지로부터 큐들 중 제 2 세트의 큐들을 선택한다. 부가 스케줄러는 어느 패킷들이 전송되는지로부터 중간 큐들을 선택한다. 다중스테이지 스케줄링은 두개 이상의 스케줄링 스테이지를 포함할 수 있으며 큐들의 각 세트는 하나 이상의 스케줄링 방법에 기초하여 스케줄링될 수 있다.

더욱 구체적으로, 본 발명은 패킷 크기가 짧을 때도(40 바이트) 매우 많은 수의 큐(바람직한 실시예에서 4096)들을 가지고 매우 높은 회선 속도(OC48 2.5Gb/s)로 플래스-기반 큐잉 및 퍼-플로(per-flow) 큐잉을 수행하는 장치를 제공한다. 단일-제거 토너먼트는 일정한 비트율 큐들 사이에서 수행된다. 상기 토너먼트의 각 매치는 최하의 카운터를 갖는 큐에 의해 획득된다. 상기 토너먼트에서 승리한 큐의 카운터가 현재 시간보다 낮으면, 상기 큐의 헤드 패킷이 전송되고 카운터는 업데이팅된다. 그렇지 않으면 개별 토너먼트가 최상으로 노력하는 큐들에 대해 실행된다.

고속 회선 속도로 패킷의 스케줄링을 허용하기 위해, 토너먼트는 상기 토너먼트를 나타내는 (키, 아이디)여러쌍의 트리를 보유하는 RAM, 현재 매칭중인 경쟁자들을 보유하는 두개의 레지스터 및 비교기를 포함하는 전용 하드웨어 구조를 이용하여 수행될 수 있다. 부가 속도 처리에 대해, 토너먼트는 중복되어 수행된다. 각각의 경우에 카운터는 변경되고, 변경된 카운터로부터 토너먼트 승자로의 경로상의 토너먼트의 매치들만이 다시 실행된다. 토너먼트의 나머지는 변경되지 않은채로 남아있다. 이런 방법으로, 토너먼트는 lg(N) 사이클로 완료될 수 있으며, N은 경쟁하는 큐들의 수이다.

상기 토너먼트를 파이프라인하면 훨씬 빠른 스케줄링을 제공하게 된다. 상기 토너먼트는 두개 이상의 파이프라인 스테이지로 분할될 수 있다. 트리의 각 레벨은 하나의 파이프라인 스테이지내에 완전히 상주한다. 스테이지 I의 최종 레벨에서의 매치가 완료되면, 그 결과는 스테이지 I+1에 전달되고, 스테이지 I는 다음 토너먼트를 자유롭게 개시한다. 최후의 경우에, 트리의 각 레벨은 개별 파이프라인 스테이지에 배

40-3

치될 수 있으며 스케쥴러는 토너먼트를 완료하고 매 클록 사이클마다 패킷을 스케쥴링할 수 있다.

일 실시예에서, 각 큐는 고정 비트율(CBR) 스케쥴링용 카운터 및 가중-적정-큐잉(WFQ) 스케쥴링용 카운터, 두개의 카운터와 관련된다. 상기 패킷 스케쥴링 엔진은 먼저 CBR 카운터를 업데이팅하기 위해 CBR 알고리즘을 이용하여 CBR 큐에 대역폭을 할당한다. 그후에, 남아있는 대역폭은 WFQ 카운터를 업데이팅하기 위해 WFQ 알고리즘을 이용하여 WFQ 큐에 할당된다. 상기 카운터와 관련된 가중치를 세팅함으로써, 주어진 큐는 CBR 대역폭 또는 WFQ 대역폭을 할당받을 수 있거나 또는 상기 주어진 큐는 먼저 CBR 대역폭의 할당량을 할당받고 모든 CBR 할당량이 만족된후에 WFQ 대역폭에 공유될 수 있다.

가변 길이 패킷의 스케쥴링을 허용하기 위해, 그리고 바이트 스터핑을 수행하는 물리 계층을 갖는 패킷 스케쥴러의 이용을 허용하기 위해, 대개 CBR 및 WFQ 스케쥴링 알고리즘은 가변 패킷 크기로 동작하고 바이트 스터핑 오버헤드의 실행 카운트를 이용하도록 증가되었다.

본 발명의 목적, 특징 및 장점은 유사 참조 부호가 도면의 용일한 부분을 나타내는 도면을 참조하여 하기의 상세한 설명으로부터 더욱 명백해질 것이다.

도면의 간단한 설명

도 1은 본 발명을 구현하는 패킷 스케쥴링 트리 구조를 도시한다.

도 2는 랜덤 액세스 메모리에서 도 1의 트리 노드 매핑을 도시한다.

도 3은 노드(105)내 카운터 값의 변화와 이에 따른 내부 노드내 변화를 가진 도 1의 트리 구조를 도시한다.

도 4는 트리 구조를 처리하는 하드웨어를 도시한다.

도 5는 도 4의 하드웨어내 트리 구조를 처리하는 순서도를 나타낸다.

도 6은 본 발명에 따라 변조되는 3개의 큐(queue)를 가진 종래기술의 시스템을 도시한다.

도 7은 각각의 리프 노드에 디스에이블 필드의 추가를 가진 본 발명의 실시예에서의 트리 구조를 도시한다.

도 8a 내지 도 8c는 파이프라인 구조로 3개의구조를 실행하는 하드웨어를 도시한다.

도 9은 연속 사이클에서의 도 9의 파이프라인의 레지스터내의 예시적인 데이터를 도시한다.

도 10은 도 9의 데이터로부터의 트리 구조를 도시한다.

도 11은 본 발명의 선택적인 실시예에서의 분류 네트워크를 도시한다.

도 12는 CBR 카운터를 업데이팅하기 위한 회로를 도시한다.

도 13은 WFQ 카운터를 업데이팅하기 위한 회로를 도시한다.

도 14는 CBR과 WFQ 스케쥴링을 사용하는 토너먼트의 순서도이다.

도 15는 카운터 값이 리프 노드내에만 저장되는 트리 구조의 선택적인 실시이예를 도시한다.

도 16은 노드(955)의 값이 변화된 후의 도 15의 트리구조를 도시한다.

도 17은 본 발명의 멀티-스테이지 실시예를 도시한다.

도 18는 도 1의 구조내에 제공되는 십진법의 노드 어드레스의 이진 표시와 관련한다.

도 19는 도 15의 트리 구조의 하드웨어 실행을 나타낸다.

도 20a와 도 20b는 도 19의 플립-플롭 어레이의 하드웨어 실행을 나타낸다.

실시예

본 발명의 바람직한 실시예가 이하 기술된다.

토너먼트 분류를 사용하는 패킷 스케쥴링

최하위(lowest) 카운터의 큐를 선택하는데 사용되는 데이터 구조가 도 10에 도시된다. 도면은 트리플(triple)의 트리를 도시한다. 각각의 트리플은 어드레스, 큐 ID 및 관련 카운터 값 또는 키를 포함한다. 트리(100-107)의 리프는 각각의 큐와 관련된 카운터를 나타낸다. 이러한 각각의 리프에 대해, 어드레스 및 큐 ID는 동일하며 키는 용일 큐와 관련된 큐의 값을 유지한다. 명료함을 위해, 도면은 큐중 8개만을 도시하지만, 바람직한 실시예에서는 4,096개의 큐가 존재한다.

도 1의 트리(108-114)의 각각의 내부 노드는 리프 노드를 따라 토너먼트내 매치(match)의 결과를 나타낸다. 예를 들면, 노드(109)는 리프 노드(102와 103) 사이의 매치의 결과를 나타낸다. 노드(102)가 하부 키를 가지기 때문에, 이 매치에서 승리(win)하고, 노드(109)는 큐 ID와 노드(102)로부터의 키를 복사함으로써 이러한 승리를 기록한다.

노드(109)가 어드레스는 트리내 자신의 위치를 나타낸다. 도 18에 도시된 바와 같이, 어드레스는 b-비트 수이고 여기서 b=lg(N)+1이다. 도 1과 도 18의 예시적인 트리에서, N=8이고 b=4이다. 바람직한 실시예에서, N=4096이고 b=13이다. 내부 노드가 최상위 비트 비트 세트를 가진 어드레스를 가지는 반면 리프 노드는 명확한 최상위 비트를 가진 어드레스를 가진다. 어드레스는 부모(parent), 자식(offspring) 및 형제(sibling)의 어드레스의 계산을 용이하게 하기 위해 할당된다. 어드레스 a를 가진 노드의 부모는 a

40-4

를 우측으로 1비트 이동시키고 최상위 비트를 설정함으로써 계산된다. 유사하게, 내부 노드 b의 윗(upper) 자식의 어드레스는 b를 좌측으로 1비트 이동시킴으로써 계산되고 아래(lower) 자식은 윗 자식의 어드레스를 가지지만 최하위 비트 세트를 가진다. 마지막으로, 노드 형제의 어드레스는 노드 어드레스의 최하위 비트를 보수(complement)함으로써 주어진다. 이러한 계산은 통상적으로 다음과 같이 'C' 프로그램 언어로 표현된다:

$parent(a)$ a | (a >> 1);

$upperChild(b)$ b << 1;

$lowerChild(b)$ (b << 1) | 1;

$sibling(c)$ c^1;

이러한 계산은 이동이 오로지 이동에만 관련하므로 하드웨어에서 특히 효율적이고, 이는 오로지 기록, 단일-비트 OR 또는 XOR 동작만을 필요로 한다.

N개의 큐로 구성된 세트가 주어지면, N개의 리프 노드를 가진 트리가 제공되고, 매치는 반드시 승자(winner)(최소 카운터를 가진 큐)를 선택하도록 토너먼트내에서 수행되어야 한다. 하지만, 토너먼트당 매치의 수를 QoS 출력 스케줄링에 대해 오로지 하나의 카운터, 최종 선택된 큐, 그 시간의 변화만을 관찰함으로써 최대 lg(N)으로 감소시킬 수 있다. 따라서, 오로지 이러한 큐를 포함하는 이들 매치만이 리플레이된다. 이들은 변화된 리프 노드로부터 트리의 루트로의 경로상의 매치이다. 이 경로에 있지 않은 다른 매치의 출력은 일정하게 남아있는다.

토너먼트의 결과를 계산하는 이러한 증가(Incremental) 방법이 도 3에 도시되어 있다. 이러한 도면은 큐(5) 이후의 트리의 상태를 도시하고, 도 1의 토너먼트에 의해 선택된 큐는 10에서 9로 증가된 카운터(및 그에 따른 키)를 가진다. 트리내 변화된 값이 굵은 글자로 표시되고 리플레이된 매치는 굵은 선으로 표시된다. 노드(105)의 키가 변하기 때문에, 노드(104와 105) 사이의 매치가 리플레이된다. 이는 노드(110)의 키를 변화시켜, 110과 111 사이의 매치가 리플레이되도록 한다. 이때 큐(5)가 이러한 매치에서 패배(lose)하여, 노드(113)의 ID와 키가 변화되도록 한다. 마지막으로, 노드(112와 113) 사이의 매치는 매치와 이에 따른 토너먼트를 통 승리한 노드(2)로 리플레이된다.

따라서, 도면이 도시하는 바와 같이, 단일 카운터가 업데이트되고난 후, 토너먼트는 전체 N-1 매신에 lg(N) 매치만을 리플레이함으로써 재계산될 수 있다. 바람직한 실시예에서, 이는 매치의 수를 4,095로부터 12로 감소시킨다.

하드웨어 패킷 스케줄링 엔진

하드웨어내 토너먼트를 수행하기 위해, 트리의 노드가 도 2에 도시된 바와 같이 RAM내 테이블에 맵핑된다. 여기서 RAM(200)은 도 1의 트리의 15개의 노드중 하나를 나타내는 기록을 각각 유지하며 0에서 14로 어드레스된 15개의 위치를 가진다. 첫 번째 8개의 위치는 8개의 나머지 노드를 나타내는 기록을 홀딩하고 나머지 7개의 위치는 도 1의 트리의 7개의 내부 노드를 나타내는 기록을 홀딩한다. 각 기록은 현재의 매치에 다다른 큐(나머지 노드)를 확인하는 ID 및 이 큐에 대응하는 키(카운터 값)를 나타내는 두 영역으로 구성된다. 각각의 나머지 노드 기록에 대해, ID는 기록을 홀딩하는 위치의 주소와 언제나 일치한다.

토너먼트를 증가적으로 실행하기 위한 하드웨어 엔진은 도4에 도시된다. 각 노드 기록의 ID 및 키 영역은 듀얼 포트 SRAM(200)에 홀딩된다. SRAM(200)은 어드레스(321)를 가진 판독 포트, 데이터 출력 라인(323), 어드레스(322)를 가진 기록 포트 및 라인(324) 내의 데이터를 가진다. 엔진은 3개의 레지스터, 즉 어드레스 레지스터(301), 비교 레지스터(312) 및 형제 레지스터(311)를 포함한다. 어드레스 레지스터(301)는 가장 최근에 업데이팅된 노드의 어드레스를 홀딩한다. 이하에서 이러한 노드는 현재 노드(current node)로 언급된다. 비교 레지스터는 현재 노드의 ID 및 키를 홀딩한다. 현재 노드(부모와 현재 노드를 공유하는 노드)의 형제 키 및 ID는 형제 레지스터에 저장된다. 이하에서 이러한 노드는 형제 노드로 언급된다.

현재 노드의 부모 및 형제를 계산하기 위한 로직은 어드레스 레지스터(301)와 관련된다. 비교 레지스터에서 업데이팅된 노드 기록이 현재 노드 어드레스에서 테이블에 다시 기록되어야 하므로, 어드레스 레지스터의 내용은 RAM(200)을 위한 기록 어드레스(322)로서 직접 사용된다. 다음 매치를 실행하기 위해, 이 노드의 형제는 인출되어야 하고, 어드레스 레지스터(322)의 내용은 RAM(200)에 대해 판독 어드레스(321)를 생성하기 위해 (마지막 신호 비트를 보수화하여) 형제 기능(302)을 통해 전달된다. 현재 노드 및 형제 노드가 비교된 후, 두 노드의 부모는 새로운 현재 노드가 되고 매치의 승자는 이러한 새로운 현재 노드의 업데이팅된 값이 된다. 이러한 스텝업 트리에 대한 어드레스를 계산하기 위해, 현재의 노드 어드레스(322)는 현재 노드 어드레스의 새로운 값을 계산하기 위해 부모 기능(303)(오른쪽으로 시프팅 및 최상위 비트를 세팅)을 통해 전달된다. 멀티플렉서(304)는 새롭게 증가된 토너먼트의 시작에서 라인(325)으로부터 남겨진 노드의어드레스를 갖는 현재 노드 어드레스를 로딩하기 위해 사용된다.

비교 및 형제 어드레스와 관련된 로직은 토너먼트를 초기화하고, 토너먼트에서 각 매치에 요구되는 키 비교를 실행하고, 각 매치의 승자의 ID 및 키로 현재 노드를 업데이팅하기 위해 사용된다. 비교기(313)는 라인(328) 상의 형제 노드의 키를 라인(329) 상의 현재 노드의 키와 비교한다. 만일 형제 키가 현재 노드 키 이하일 경우, 라인(331)이 요구되고 비교 레지스터가 다음 사이클의 시작에서 형제 레지스터의 내용으로 로딩된다.

멀티플렉서(315)는 각 토너먼트의 시작에서 라인(325)으로부터 업데이팅된 큐의 ID를 갖는 비교 레지스터(312)의 ID 영역을 로딩한다. 이러한 큐 ID는 또한 남겨진 노드 어드레스임을 주의해야 한다. 따라서, 동일한 값이 토너먼트의 시작에서 어드레스 레지스터(310)로 로딩된다. 멀티플렉서(316)는 각 토너먼트

40-5

의 시작에서 큐 카운터의 업데이팅된 값을 갖는 비교 레지스터(312)의 키 영역을 로딩한다.

토너먼트 소트 연산은 도5의 흐름도에서 설명된다. 소트는 비교 레지스터(312) 및 어드레스 레지스터(301)가 초기화 되는 단계(401)에서 시작한다. 카운터가 업데이팅된 큐의 ID는 멀티플렉서를 통해 어드레스 레지스터(301) 및 멀티플렉서(315)를 통해 비교 레지스터의 ID 영역으로 로딩된다. 동시에, 이러한 큐와 관련한 카운터의 값은 멀티플렉서(316)를 통해 비교 레지스터(312)의 키 영역으로 로딩된다.

일단 토너먼트가 초기화되면, 각 스텝업 트리는 단계(402 내지 405)로 구성된 루프의 1회 반복으로 구성된다. 비록 이러한 루프가 네 단계를 포함하지만, 각 반복은 단일 클록 주기에서 발생한다. 우선, 단계(402)에서, 비교 레지스터의 업데이팅된 현재 노드는 RAM 내의 적절한 어드레스에 저장된다. 이러한 기록은 라인(322) 상의 기록 어드레스를 사용하여 라인(324)의 데이터에 대해 발생한다. 동시에 이러한 노드의 형제에 대한 노드 기록은 라인(321) 상의 형제 어드레스를 사용하여 RAM(200)으로부터 판독된다. 형제 기록은 데이터 아웃 라인(323) 상에 나타나며 형제 레지스터(311)에 로딩된다. 레지스터는 형제 레지스터의 출력이 동일한 주기에서 형제 기록이 판독되는 동안 형제 레지스터의 출력이 비교기에 유용하도록 투명 래치이다. 당업자는 엔진을 실행하기 위해 사용된 로직 기술에 의존하여, 형제 레지스터가 생략될 수 있고 형제 기록이 RAM(200)의 출력 라인(323)으로부터 직접 액세스된다는 것을 알 것이다. 또한 단계(402)에서, 현재 노드 어드레스의 부모가 계산되고 이러한 값이 주기의 마지막에 어드레스 레지스터에 로딩되어 다음 주기에 RAM(200)에 의해 사용된다.

RAM(200)이 판독 및 기록된 후, 형제 및 현재 노드가 비교되고 이에 따라 현재 노드가 업데이팅된다. 이러한 프로세스는 단계(403 및 404)에 설명된다. 결정 단계(403)에서, 형제 레지스터(328)의 영역 및 비교 레지스터(329)의 키 영역이 비교기(313)에 의해 비교된다. 만일 단일 키가 현재 노드 키 이하일 경우, 흐름 단계는 단계(404)로 전행하고 비교 레지스터는 주기의 마지막에 형제 레지스터로부터 업데이팅된다. 단계(404)는 형제가 매치를 승리하여 새로운 현재 노드가 된 경우를 나타낸다. 형제 레지스터로 비교 레지스터를 로딩하는 것은 부모 노드가 매치의 결과를 기록하는 다음 주기에 승리 ID 및 키로 기록되도록 한다. 만일 현재 노드 키가 형제 노드 키 이하일 경우, 현재 노드는 매치를 승리한다. 이 경우, 흐름은 단계(404)를 스킵하고 비교 레지스터는 변화되지 않은 채로 남는다. 이는 부모 노드가 다시 매치의 결과를 기록하는 다음 주기에서 현재 노드의 값으로 기록되도록 한다.

단계(405)는 토너먼트의 완료를 체킹한다. 만일 업데이팅 후, 어드레스 레지스터가 테이블의 루트 위를 나타낼 경우, 토너먼트는 완료되고 승리 큐의 ID는 비교 레지스터(312)의 ID 영역에 유지되며 라인(330)에 유용하다. 본 발명의 예로든 8개의 큐에서, 어드레스 레지스터가 어드레스 14에서의 루트 노드를 1 넘는 값 15로 업데이팅될 때, 이러한 상태가 발생한다. 통상적으로, 이러한 종결 상태는 2진수로 나타나는 현재의 노드 어드레스가 모두 1일 때마다 검출된다. 만일 토너먼트가 완료되지 않았다면, 흐름은 다시 노드(402)로 복귀하며, 동일한 프로세스가 반복되며 트리가 한 레벨 업된다.

빈 큐의 디스에이블스케줄링

큐가 비어있는 경우, 토너먼트에서 경쟁으로부터 디스에이블되어야 하는데, 이는 승리할 경우 출력될 패킷을 제공할 수 없기 때문이다. 이는 가장 큰 가능한 값에 카운터를 세팅함으로써 달성된다. 그러나, 이러한 해결책은 카운터 값이 도달하는 다음 패킷이 큐로부터 분리될 경우를 결정하도록 보존되야 하기 때문에 실행가능하지 않다. 도7에 도시된 실행가능한 해결책은 토너먼터 트리의 나머지 노드에 단일 비트의 디스에이블 영역을 부가하는 것이다. 이러한 비트가 세팅될 때, 노드(node)는 마치 키(key)가 최상위 값(the least possible value)(2진 레프리젠테이션의 모든 첫번째)을 갖는 것처럼 처리(treat)된다. 패킷(packet)이 빈 큐(empty queue)에 도달(arrive)할 때, 비워지지 않게(non-empty) 하고, 디스에이블 필드(disable field)가 클리어(clear)되며 증분 토너먼트(incremental tournament)는 보호성 카운터 값을 사용하여 트리의 상태(state)를 업데이트하도록 수행된다.

도 7은 도 3의 트리에서 디스에이블한 큐(7)의 예를 도시한다. 큐(7)에 해당하는, 리프 노드(leaf node)(107)는 자체의 디스에이블 비트 세트를 가지며 이로 인해 리프 노드(106)에 대하여 매칭(match)되지 않는다. 이러한 매칭의 결과는 인터미디에잇(intermediate) 노드(111)에 기록되고 이러한 결과는 트리를 번식시키며(propagate up), 인터미디에잇 노드(113)를 업데이트한다. 만약 리프 노드(107)의 디스에이블 비트는 트리의 임의의 다른 상태가 변하기(change) 전에 클리어되고 노드(107과 106) 사이의 매칭이 액츄얼 카운터 값을 사용하여 리런(rerun)된다. 다음으로 이러한 매칭의 결과는 도 3의 상태에서 귀환되는(retune) 트리를 번식시킨다.

파이프라인 패킷 스케줄링 엔진

매우 높은 라인율(line rate)에서, 도 4의 하드웨어를 사용하여 매번 3 주기(cycle)에서 하나의 증분 토너먼트(또는 4096큐에 대한 매번 12 주기에서 하나의 증분 토너먼트)를 완료(complete)하는 것이 충분하지 않다. 패킷 스케줄링의 높은 비율에서, 주기당 하나의 패킷은 도 8a-c에 도시된 것과 같은 토너먼트 소트(sort)를 파이프라이닝하여 달성될 수 있다. 도 8a-c의 회로는 세 개의 파이프라인 스테이지(stage)(600,620,640)로 구성된다. 각각의 파이프라인 스테이지는 도 4의 모든 논리 토너먼트 소트 엔진을 포함하지만, 트리의 한 레벨상에서만 동작한다. 노드 상태를 홀딩하는 테이블(200)은 세 개의 파이프라인 스테이지로 파티션된다. 로케이션(location)(0-7)인, 트리의 리프들<leaves:leaf의 복수>은 파이프라인 스테이지(600)의 RAM(610)내에 홀딩된다. 로케이션(8-11)인, 내부 노드의 제 1 레벨은 파이프라인 스테이지(620)의 RAM(630)내에 홀딩되고, 로케이션(12-13)인, 내부 노드의 제 2 레벨은 파이프라인 스테이지(640)의 RAM(650)내에 홀딩된다. 위닝(winning) 노드 ID와 키(key)가 각각 라인 (653,654)상으로 출력되는 것처럼, 위치(14)인, 트리의 루트는 저장할 필요는 없다.

예컨대, 도 3에서 수행되는 것처럼, 도 1에 도시된 초기(initial) 상태로 시작하고 큐(5-9)에 해당하는 카운터를 변환한다고 가정하자. 도 8a-c의 파이프라인 회로는 3 주기에서 이러한 변화에 해당하는 증분 토너먼트를 계산한다. 제 1 주기의 시작에서, 어드레스 레지스터(601)는 라인(690)상의 리프 노드(5)의 어드레스 ID로 로딩(load)된다. 동시에, 비교 레지스터(602)의 ID 부분이 같은 값을 가지고 로드되고, 비교 레지스터(602)의 키 부분은 라인(691) 상의 큐 카운터의 새로운 값인 9를 가지고 로드된다. 첫번째 주기동안 자식 노드 5에 대한 새로운 입력인 5.9는 기록 어드레스 포트(606)와 기록 데이터 포트(604)를

40-6

사용하여 램(610)내의 위치 5에 기록된다. 또한, 이 주기동안 이 노드의 현재 노드인 노드 4에 대한 레코드는 판독 어드레스 포트(607)와 판독 데이터 포트(605)를 사용하여 램(610)으로부터 판독된다. 예에서, 이 레코드의 이 값은 4.10이다(ID = 4, key = 10). 라인(607) 상의 판독 어드레스 4는 라인(606) 상의 현재 노드 어드레스 5의 최하위 비트를 보충하는 형제 유닛(615)을 사용하여 생성된다. 당 분야의 숙련된 자라면 만약 램(610)이 정적인 출력을 제공한다면, 현제 레코드의 값은 램 출력 포트(605)로부터 직접적으로 접근될 수 있고 어떠한 형제 레지스터도 요구되지 않는다는 것을 이해할 것이다. 어느 경우에나, 라인(608) 상의 형제 레코드 10의 키 필드는 비교기(617)에 의해 라인(609) 상의 현재 레코드 9의 키 필드와 비교된다. 각각 10와 키 필드를 위하여, 비교 레지스터(602)로부터 현재의 레코드 5.9를 선택하거나 라인(613, 614) 상의 다음 스테이지에 출력이 될 형제 레지스터(603)으로부터 형제 레코드 4.10을 선택하기 위해, 비교기(661)의 출력은 멀티플렉서들(618, 619)을 제어한다. 이 경우에 있어서, 9는 10보다 작고, 그래서 비교기는 비교 레지스터로부터 출력으로 5.9를 선택하게 된다. 이 선택과 병행하여, 부모 로직(616)은 라인(606) 상의 현재의 어드레스를 오른쪽으로 시프트(shift)하고 최상위 비트를 세팅(setting)함으로써 다음 스테이지 10에서 접근될 노드의 어드레스를 계산한다. 이 어드레스는 라인(612) 상의 다음 스테이지에의 출력이다.

두번째 주기동안, 파이프라인(620)의 두번째 스테이지는, 트리의 두번째 랭크에 있는 내부 노드(110)를 업데이트하고 유사한 방법으로 노드 110과 111 사이의 경기의 승자를 계산한다. 이 주기의 시작점에서, 라인(612) 상의 노드 어드레스 10은 어드레스 레지스터(621)로 로드되고, 라인(613, 614) 상에 있는, 마지막 스테이지로부터의 승자 노드 레코드 5.9는 비교 레지스터(622)로 로드된다. 그 다음에 램(630)은 형제 레코드 7.6을 판독한다. 비교기는 형제와 현재의 레코드들의 키를 비교하고 라인(633, 634) 상의 출력이 될 가장 낮은 키를 갖는 레코드를 선택한다. 이 경우에서 형제 레코드 7.6은 출력이 된다. 동시에, 부모 로직(636)은 노드(10)의 부모의 어드레스를 계산하고, 이 경우에, 노드(13)은 라인(632) 상에 이 값을 출력한다.

최종의 파이프라인 단계(640)은 유사한 방식으로 동작한다. 그것은 노드 어드레스(13)를 받아들이고, 그것의 새로운 레코드, 7.6으로 노드를 업데이트하며, 그것의 형제 레코드, 2.5를 읽고, 키를 비교한 다음, 위너(winner), 2.5를 출력한다.

도 8A-C의 파이프라인된 스케쥴링 엔진(pipelined scheduling engine)은 하나의 증분 토너먼트를 동일한 시간 내에 도 4의 반복 엔진(iterative engine)으로서 계산한다. 그러나, 그것의 이점은 그것이 각 주기마다 새로운 증분 토너먼트를 시작할 수 있다는 것이다. 이 파이프라인된 동작의 예는 도 9의 표에 나타난다. 도면은 시간 함수로서 다른 열(row)상에서 회로 안에 있는 각 레지스터의 내용물(content)을 보여준다. 각각의 타임스텝(timestep)은 분리된 칼럼에 나타난다. 도면은 세 개의 증분 토너먼트 계산을 보여주는데: 큐 50에서 9까지의 카운터를 변화시키고(도 3에 나타남), 사용중지(disabled) 큐 7을 표시하며(도 7에 나타난대로임, 도 9에서 사용중지된(disabled) 상태는 d로 표시됨), 마지막으로 큐 20에서 15까지의 카운터를 변화시킨다(도 10에 나타남).

각각의 이 토너먼트는 연속하는 주기로 시작되는데: 주기 1에서 5.9, 주기 2에서 7.d, 주기 3에서 2.15이고, 각각의 토너먼트는 완성되기 위해 세 개의 주기를 취한다. 주어진 토너먼트는 각각의 주기 동안 하나의 파이프라인 단계에서 다음 단계로 진행한다. 도 9에서의 굵은 선은 토너먼트 5.9에 대한 진행을 보여준다. 이 토너먼트는 주기 2에서 제 2 파이프라인 단계로 진행하기 때문에, 이 실시예에서 제 1 파이프라인 단계는 다음 토너먼트, 7.d를 시작하기 위해 제거된다. 유사하게 제 1 토너먼트는 주기 3의 초기에 제 2 파이프라인 단계를 제거하고, 제 2 토너먼트가 그 주기동안 제 2 단계로 진행하게 한다. 이러한 방식으로, 파이프라인된 조직(pipelined organization)은 새로운 토너먼트가 각 주기마다 시작될 수 있도록 하는데 이 때 각 토너먼트는 그것이 시작한 후에 세 개의 주기를 완성한다. 4096 큐를 가진 바람직한 실시예에서, 파이프라인은 12단계를 갖고 각 토너먼트는 그것이 시작한 후에 12 주기를 마친다.

당업자는 도 4의 반복 엔진(iterative engine)과 도 8A-C의 풀리파이프라인된 엔진(fully-pipelined engine)이 설계 가능성의 연속체(continuum) 상에서 두 개의 극점이라는 것을 이해할 것이다. 예를 들어, 4096 큐로, 당업자는 양자택일로 여섯 단계 파이프라인 사용할 수 있으며 상기 파이프라인에서 각 단계는 트리(tree)의 두 개 레벨을 다루고 그것의 결과를 다음 단계로 넘기기 전에 두 번의 반복을 수행한다. 또한 다른 조합(예를 들어, 각각 3회의 반복을 수행하는 4개 스테이지, 각각 4회의 반복을 수행하는 3개의 스테이지, 또는 각각 6회의 반복을 수행하는 2개의 스테이지)가 성능에 대한 실시 복잡도가 교환될 수 있다.

정렬 네트워크를 사용하는 패킷 스케쥴

고속 패킷 스케쥴의 또다른 방법은 도 11에 도시된 것처럼 정렬 네트워크를 사용한다. 정렬 네트워크는 그의 입력 라인(711-718) 상에 8개의 큐 ID/키 쌍들을 허용하고 출력 라인(721-728) 상의 내림순서 키 크기 순서대로 이를 레코드를 정렬한다.

8개의 입력에 대해, 정렬 네트워크는 각각 4개의 비교 교환 유니트를 포함하는 6개의 스테이지(702-707)로 구성된다. 일반적으로, 2^n개의 입력을 갖는 네트워크는 각각 2^{n-1}개의 비교 교환 유니트를 포함하는 2n개의 스테이지를 필요로 한다. 굵은선으로 연결된 2개의 서클은 각각 비교 교환 유니트, 즉 701을 나타낸다. 이러한 유니트는 그의 좌측 단자상에 2개의 입력을 허용하고, 2개의 입력의 키 필드를 비교하며, 필요한 경우 입력을 교환하여 레이저 키를 갖는 레코드는 상부 우측 단자상으로 출력되고 보다큰 키를 갖는 레코드는 하부 우측 단자상으로 출력된다.

정렬 네트워크가 각각의 사이클의 새로운 세트의 업데이트를 처리하며, 주어진 시간 사이클에서 그의 비교 및 교환을 완성하는 정렬 네트워크의 각각의 스테이지를 사용하며, 도 8A-8C의 토너먼트 검색 엔진과 동일한 방식으로 파이프라인 될 수 있다. 또한, 하드웨어 복잡도를 감소시키기 위해, 정렬 네트워크는 각각의 스테이지에 대해 단지 하나의 비교 교환 유니트만을 사용하고 스테이지에서 교환된 입력 쌍에서만 작동하게 함으로써 증분방식으로 업데이트될 수 있다.

당업자는 도 11의 정렬 네트워크가 8개 입력의 일괄 정렬 네트워크임을 알 것이다. 이러한 네트워크는

40-7

다수의 입력으로, 예를 들어, 2048개의 비교 교환 유니트의 24개 스테이지를 갖는 4096-입력 네트워크로 확대될 수 있다. 또한 다른 정렬 네트워크(예를 들어, Chapter 28, Cormen, Leiserson, Rivest, Introduction to Algorithms, MIT Press, 1990)는 일괄 네트워크로 대체될 수 있다.

정렬 네트워크를 사용하는 스케쥴러는 토너먼트 트리를 사용하는 스케쥴러보다 고도의 하드웨어 복잡도 및 고도의 성능을 갖는다. 정렬 네트워크를 사용하여 멀티플 패킷은 정렬 네트워크의 상부 몇개의 출력을 선택함으로써 단일 사이클에서 스케쥴처리될 수 있다. 또한 몇개의 큐 카운터는 완전 정렬이 각각의 업데이트상에서 수행됨에 따라, 토너먼트 트리상에서 수행된 부분 정렬보다는 단일 사이클에서 업데이트된다.

조합 CBR 및 WFQ 스케쥴

바람직한 실시예에서, 각각의 큐는 2개의 카운터와 연관된다. 제 1 카운터 STT는 고정 속도형 서비스(CBR; constant bit rate) 안내설명에 따라 패킷을 위한 스케쥴 전송 시간을 보유한다. 제 2 카운터 TTT는 가중-페어-큐잉(WFQ; weithted-fair-queuing)을 사용하여 최상 트래픽에 대한 최상 베이시스 상에 완벽한 패킷을 위한 이론적 전송 시간을 보유한다. 관련된 각각의 카운터는 도 7에 도시된 것처럼 카운터가 토너먼트에 참여하지 못하도록 하는데 사용될 수 있는 디세이블 비트이다. 따라서, 주어진 큐동안, WFQ 디세이블 비트가 설정된다면, 큐에서의 패킷은 CBR 서비스 안내설명에 따라 스케쥴된다. CBR 디세이블 비트가 설정된다면, 큐에 있는 패킷은 모든 CBR 패킷이 WFQ 정책을 사용하여 스케쥴된 후에 나머지 임의의 대역폭과 시합된다. 모든 디세이블 비트가 소거된 경우, 패킷은 CBR 서비스 안내설명에 따라 먼저 스케쥴된다. 다음, 모든 CBR 트래픽을 만족시킨 후에 임의의 대역폭이 남는 경우, 큐에서의 패킷은 WFQ 정책을 사용하여 남는 대역폭에 대한 최상 트래픽과 시합된다. 마지막으로, 모든 디세이블 비트가 설정되는 경우, 큐는 중단되어 어떠한 패킷도 스케쥴되지 않는다.

이러한 이중 스케쥴 정책을 실시하기 위해 2개의 토너먼트 트리가 유지된다. 한개의 트리는 키로서 각각의 큐에 대한 STT 카운터를 사용하며 다른 트리는 키로서 TTT 카운터를 사용한다. 각각의 스케쥴 간격동안, 전송되는 다음 패킷이 도 14의 순서도에 도시된 것처럼 선택된다. 프로세스는 CBR 토너먼트의 승자가 검사된는 박스(901)에서 개시된다. 승자의 키는 WSTT(승자의 STT)로 표시되며 큐 ID는 WCID(CBR 토너먼트 ID의 승자)로 표시된다. WSTT가 현재 시간과 같거나 또는 그 이하인 경우, 승리한 CBR 대기열에서의 패킷은 전송을 무시하고 제어가 이 패킷을 전송하도록 박스(903-905)에서 이루어져 스케쥴 상태에 따라 업데이트된다. WSTT가 현재 시간보다 큰 경우, CBR 패킷은 최상 패킷이 선택되는 박스(906-909)에서 적절한 전송 및 제어가 이루어질 수 없다.

WSTT가 현 시간보다 짧거나 동일한 경우에, 큐 WCID 헤드의 패킷 전송은 박스(903)에서 초기화된다. 이 패킷은 가장 빠른 STT를 가지는 CBR 패킷이며, 이에따라 라인을 통해서 전송되는 제 1 패킷이여야 한다. 패킷 전송이 시작된 후에, 이 큐에 대한 STT는 박스(904)의 이 패킷 전송에 의해서 소모되는 대역폭 자원을 반영하도록 업데이트된다. 이 패킷에 대한 새로운 STT는 NSTT로 표시된다. 이 전산화의 세부사항은 도 12에 도시되고, 하기에서 상세히 기술된다. 박스(905)에서 CBR 토너먼트는 큐 WCID에 대한 새로운 STT를 반영하도록 업데이트된다. 이 업데이트는 도 4의 반복 회로 또는 도8A-C의 파이프라인 회로를 사용하여 이루어질 것이다. 일단 증가된 토너먼트가 완성되고, 패킷의 전송이 완성된다면, 제어신호는 어떤 패킷이 다음에 전송될 것인지를 결정하도록 박로(901)로 되돌려진다.

현 시각에에서 CBR 패킷이 전송에 적합하지 않다면, WFQ 스케쥴은 WFQ 토너먼트의 승자가 결정되는 박스(906)에서 시작된다. 승자의 키는 WTTT(승자의 TT)를 의미하고, 승자의 ID는 WWID(WFQ토너먼트 승자의 ID)를 의미한다. 승리 WFQ 패킷이 박스(907)로 전송된다. 이것은 WFQ 스케쥴 알고리즘에 따른 유용한 대역폭의 다음의 유용한 부분을 얻도록 계획된 최선의 패킷이다. 박스(908)에서, 승리 큐에 대한 TTT가 업데이트된다. 이 TTT 업데이트 전산화의 세부사항은 도 13에 도시되고 하기에서 기술된다. 증가된 토너먼트는 큐 WWID에 대한 새로운 TTT를 가지는 트리를 업데이트하기 위해서 박스(909)의 WFQ 트리에서 수행된다. 최종적으로 토너먼트 및 패킷 전송이 완성된 후에, 제어신호는 다음 패킷을 계획하는 박스(901)로 되돌려진다.

동일한 하드웨어가 WFQ 토너먼트 및 CBR 토너먼트를 수행하는 데 사용될 수 있다. 소정의 스케쥴링 동안에, 하나의 토너먼트 또는 또다른 토너먼트가 수행된다. 두 토너먼트들을 동시에 수행할 필요는 없다. 동일 소팅 엔진에서 두 토너먼트를 지지하기 위해서, 각 RAM(도 4의 200 또는 도 8A-C의 610,630 및 650)는 두 토너먼트에 대한 트리를 유지하도록 크기가 두배가 되고, 고속의 주소 비트는 활성 토너먼트를 선택하는 데 사용된다.

바이트 스터핑으로의 다양한 패킷 길이 스케쥴링

도 12는 CBR 서브 연결과 관련하여 전송되는 패킷이 전송된 후에 새로운 STT를 전산화하는 데 사용되는 로직을 도시한다. 이 로직은 다양한 길이 패킷 및 비둘리큐에 의한 비트-스트핑을 조정하도록 증대된 ATM '리키-버킷(leaky-bucket)' 트래픽 셰이핑 알로리즘의 로직을 뒤따른다. 로직은 하부트랙터(802)에서 현재시각에서의 프리셋 한계를 추출함으로써 시작된다. 라인(803)에서의 이러한 추출 결과는 최대 유니트(804)의 라인(801)에서의 현 STT와 비교되고, 이 두개중 보다 큰것이 새로운 STT가 계산되는 베이스라인으로서 사용되도록 라인(805)상으로 출력된다. 현시각 전에 프리셋 한계 이상으로 이 베이스라인을 한정시킴으로써, 주기동안의 사용되지 않는 플로우는 현재 시간을 따라갈 때까지 출력라인을 과도하게 사용하는 것을 막는다. 효과면에서, 플로우의 과도한 증가는 한계에 대한 CBR 웨이트의 비율로 한정된다. 이 비율은 사용되지 아니하고 현시각을 따라가는 데에 소비되는 동안, 큐가 쌓이는 정도를 의미하는 비트수를 나타낸다.

라인(805)상의 베스트라인은 모든 오버헤드를 포함하는 현 패킷을 일정한 비트속도로 전달하는 데 걸리는 시간에 따라 증가된다. 첫째, 오버헤드의 두가지 형태는 라인(807) 상에 전체 패킷 길이를 산출하도록 가산기(806)의 패킷 길이에 첨가된다. 오버헤드의 첫번째 오버헤드는 헤드기 및 프레밍 정보에 대한 패킷 당 요구되는 일정 오버헤드이다. 오버헤드의 두번째 오버헤드는 몇몇의 물리층에 의해 요구되는 바이트 스트핑 때문에 오버헤드되는 물리층이다. 각각의 큐는 각 패킷 전송시 스트핑되는 바이트 수를 카운트하고, 이 카운트된 값을 다음 패킷이 전송될 때 두번째 오버헤드 형태로서 첨가한다. 이 카운트를 계

속하는 것은 심지어 소정의 패킷에 대해 스트핑되는 바이트 수가 패킷이 전송될 때까지 알려지지 않는다 할지라도 바이트 스트핑이 정확히 큐에 대한 CBR 계산시에 고려된다.

그런후 라인(807) 상의 전체 패킷 길이는 캐릭터 비트 속도로 모든 오버헤드를 갖는 패킷을 전송하는 데 요구되는 시간 간격을 산출하도록 멀티플라이어(808) 내의 CBR 웨이트에 의해서 증가된다. CBR 웨이트는 플럭 트렁크 비트 속도의 일부로서 캐릭터 비트 속도의 역수이다. 즉, CBR 가중치(Weight) = 플럭 트렁크 비트율/고정 비트율이다. 시간 간격이 멀티플라이어(808)에 의해 계산되면, 가산기(809)에 의해 베이스라인 STT에 가산되고, 그 결과 생기는 새로운 STT는 퍼-큐 레지스터(per-queue register)(800)에 저장된다.

가중 헐핑 대기정렬(weighted-fair-queuing)을 사용한 예정된 대기정렬을 위한 새로운 TTT의 계산은 도 13에 도시된다. 계산은 상기에서 상술된 CBR 알고리즘을 위한 것과 유사한 방식으로 가변 패킷 길이와 패킷 오버헤드를 다루기 위해 논의된 가중 헐핑 대기정렬 알고리즘(참고로, Zhang에 의해 1995년 10월 IEEE 83(10)에 게시된 '패킷 스위칭 네트워크에서 보장된 성능 서비스를 위한 서비스 효련')과 유사한 방식으로 진행된다. 오버헤드는 가산기(820)에 부가되며, 그리고 전체 패킷 길이는 멀티플라이어(822)내의 WFQ 가중에 의해 곱해진다. 멀티플라이어(822)의 출력은 가산기(824)에 의해 베이스라인 TTT에 가산되며, 새로운 TTT는 퍼-큐 레지스터(826)에 저장된다.

대안적인 토너먼트 소트(Alternative Tournament Sort)

토너먼트 소트 알고리즘의 대안적인 실시예는 트리의 각 내부 노드에서 각각의 경기의 결과를 저장하는 단계를 포함한다. 이것은 각 내부 노드에서 경기 승자의 ID와 키를 저장하는 것과 대조를 이룬다. 이러한 대안적인 실시예는 도 15와 16에서 도시된다. 도 15는 각 내부 노드에서 저장된 각각의 경기의 결과를 제외하고는, 도 1과 같이 동일한 토너먼트를 나타내는 토너먼트 트리를 도시한다. 예를 들면, 잎 노드(leaf node)(950, 951)간의 경기는 상부 노드(950)에 의해 승리되어, 0은 내부 노드(958)에 저장된다. 내부 노드(958, 959)간의 경기는 하부 노드(959)에 의해 승리되어, 101 내부 노드(962)에 저장된다.

도 15의 트리의 임의의 노드와 관련된 ID와 키는 노드로부터 승자 잎의 경로를 추적함으로써 확인될 수 있다. 경로의 각 단계에서, 만약 (위쪽 어린이가 경기를 승리했다는 것을 나타내는) 노드가 0를 유지하면, 위쪽 어린이가 선택되며, 만약 (아래쪽 어린이가 경기를 승리했다는 것을 나타내는) 노드가 1를 유지하면, 아래쪽 어린이가 선택된다. 예를 들면, 토너먼트 승자과 관련된 ID와 키, 노드(964)는 트리내에서 ID(5)와 키(1)를 갖는 노드(955)에 굵은 경로를 추적함으로써 확인된다. 이러한 경로를 따라 횡단된 내부 노드의 상태는 이러한 노드 101 - 5의 ID를 제공하기도 한다. 임의의 경기의 승자에 상응하는 ID는 특정 경로를 따라 적절한 내부 노드 값을 선택하기 위하여 멀티플렉서를 사용함으로써 신속히 생성될 수 있다.

잎 노드가 갱신될 때, 토너먼트 소트는 상술된 바와 같이 유사한 방식으로 증분적으로 재계산된다. 예를 들면, 노드(955)와 관련된 카운트나 키가 1에서 9로 변화되는 도 16내에 도시된 경우를 고려해보자. 노드(955)에서 뿌리 노드(964)로의 경로를 따르는 각각의 경기는 차례대로 재계산되어야 한다. 우선, 제 1 레벨 경기를 위한 경쟁자의 ID는 5의 LSB를 컴플리멘팅함으로써 계산된다. 그리고 나서, 노드(955)의 키(9)와 노드(954)의 키(10) 사이의 경기가 진행되며, 955가 승자가 된다. 내부 노드(960)는 이러한 경기 결과를 기록하기 위하여 1에 업데이트된다. 그 다음에, 제 2 레벨 경기를 위한 경쟁자의 ID는 960의 형제, 노드(961)에서 ID(7)과 키(6)를 갖는 승리한 잎 노드(957)로의 경로를 추적함으로써 계산된다. 여기서, 957은 경기를 이기며, 그래서 내부 노드(963)는 이러한 결과를 기록하기 위하여 1로 갱신된다. 최종 경기를 위한 경쟁자는 963의 형제, 노드(962)에서 승리한 잎 노드(952)로의 경로를 추적함으로써 확인된다. 이러한 경우에, 952는 경기를 이기며, 그래서 뿌리 노드(964)는 이러한 결과를 기록하기 위하여 0으로 갱신된다. 경기의 경쟁에서, 964에서 잎 노드(952)로의 경로는 010 값을 가지며, 잎 노드(2)의 ID는 2진수이다.

도 4와 8의 하드웨어 엔진은 각 경기의 결과만이 트리에 저장되는 이러한 대안적인 토너먼트 탐색에 쉽게 적용될 수 있다. 이러한 적용은 단지 잎 노드만을 유지하기 위하여 RAM의 크기를 감소시키며, 내부 노드의 값을 유지하기 위하여 플립-플롭 어레이를 부가시키며, 그리고 상기 플립 플롭의 상태에서 경기 승자의 ID를 계산하기 위하여 멀티플렉서를 부가함으로써 실행된다. RAM은 토너먼트의 개시에서만 기록되므로, 개별 기록 포트를 구비할 필요가 없다. 모든 일련의 상태 변화는 플립-플롭 어레이내에 기록된다. 도 8a-c의 파이프라인 실행에서, 각각의 파이프라인 단내의 개별적인 RAM은 각각의 파이프라인 단을 위한 개별적인 판독 포트를 갖는 단일의 멀티-포트 RAM에 의해 대체되어야만 한다.

도 19는 이러한 대안적인 토너먼트 소트를 실행하는 파이프라인이 아닌 하드웨어 엔진의 블록도를 도시한다. 이것은 도 4의 논리의 변형이며, 상응하는 성분은 동일한 참조번호가 사용된다. 그러나, 이러한 경우에서 RAM(200)은 단지 잎 노드의 반대 값만을 유지한다. RAM은 내부 노드를 위한 ID와 상태를 저장하지 않는다. 소트의 시작시, 하나의 큐에 대한 카운터가 값을 변화시킬때, 변화된 큐의 ID 및 키(카운트)는 라인(325 및 326)상의 엔진에 입력된다. 제 1 사이클동안, 상기 키는 이 큐에 대한 저장된 키를 업데이트하기 위하여 RAM 어레이에 기록된다. 각각의 추후 사이클동안, 어드레싱 로직은 이전 사이클 동안 어드레스된 노드의 부모 어드레스를 303에서 계산하며 트리를 루트쪽으로 가게한다. 이 노드 어드레스는 트리의 레벨에서 현제 위너인 리프 노드의 ID를 레지스터(992)에 차례로 리턴하는 플립 플롭 어레이(991)의 멀티플렉서에 인가된다. 그 다음 형제 위너 어드레스는 내부 노드에서 위너 형제에 해당하는 키를 레지스터(311)에서 판독하기 위하여 RAM(200)에 인가된다. 이 키는 현재 노드 키와 비교기(313)에서 비교된다. 이 비교 결과는 플립 프롭 어레이 상태를 업데이트하고, 만약 새로운 키가 위너이면, 현재 키 레지스터(312)의 키를 대체한다.

도 19의 플립 플롭 어레이(991)의 상세한 항목은 도 20A-B에 도시된다. 플립 플롭 어레이는 라인(n3 내지 n0)상에 현재 노드 어드레스를 받아들이고 라인(w2 내지 w0)상에서 형태 토너먼트를 위하여 위닝 리프 노드의 어드레스를 출력한다. 인버터(1010)는 LSB를 보상함으로써 노드 어드레스를 형제 어드레스로 전환한다. 플립 플롭(958 내지 964)은 토너먼트의 각각의 비교 결과를 홀딩한다. 플립 플롭은 도 15와 정확하게 동일한 상태로 도시된다. 멀티플렉서(1001 내지 1007)는 플립 플롭의 내용 및 입력 형제 어드레

40-9

스의 내용을 바탕으로 위너 어드레스를 계산한다. 선택적인 박스(1008 및 1009)는 토너먼트 스테이지를 바탕으로 위너 어드레스의 비트를 계산하기 위하여 멀티플렉서(1005 및 1006)를 제어하도록 사용된다.

도 20A-B의 플립 플롭 기능은 일시예에 의해 가장 잘 이해된다. 이전 위너인 노드 5가 새로운 값 9로 업데이트된 도 15에 도시된 경우를 고려하여, 토너먼트는 도 15의 왼쪽에서 오른쪽으로 리턴되어야 한다. 제 1 매칭을 위한 위닝 어드레스를 계산하기 위하여, 노드 5, 4의 형제 어드레스는 라인(n3 내지 n0)에 인가된다. 라인(n3)이 영이기 때문에, 모두 3개의 멀티플렉서(1005 및 1007)에 대한 제어 입력은 영이고 이들 멀티플렉서는 노드(4)를 통하여 출력부로의 최우측 입력을 선택한다. 그래서, 새로운 토너먼트의 제 1 매칭은 값이 키 레지스터(312)에 이미 있는 리프 노드(5), 및 값이 RAM으로부터 판독되는 리프 노드(4) 사이에 있을것이다. 노드(5)가 노드(4) 보다 선행하기 때문에 이런 매칭 (1)의 결과는 플립 플롭(960)을 업데이트하기 위하여 사용된다.

제 1 매칭이 완료된후, 부모 로직(303)은 이진수로 노드(5, 10 또는 1010)의 부모 어드레스를 계산하고, 이 어드레스를 플립 플롭 어레이에 제공한다. 이 경우, 두개의 선택 박스는 n3이 1이지만 n2가 0이기 때문에 1을 출력하고, 이것은 n1이 w2로 통과되고 n0의 보수가 w1으로 통과되게 한다. 따라서, 위너 어드레스의 상부 두개의 비트는 11일것이다. 멀티플렉서(1007)의 제어 입력은 1이고, n3로부터 그것이 비트 w0를 구동하도록 멀티플렉서(1002)의 출력을 선택하게 한다. 멀티플렉서(1002)는 차례로 앞인 비트 w2에 의해 제어되고 따라서 멀티플렉서(1004)의 출력을 선택한다. 이 멀티플렉서는 차례로 역시 1인 비트 w1에 의해 제어된다. 따라서 1인 플립 플롭(961)의 상태는 비트 w0를 구동하기 위하여 멀티플렉서(1004, 1002 및 1007)를 통하여 선택된다. 이것은 노드(961)에 의해 제공된 매칭 위너가(어드레스 11에서) 노드 7이기 때문이다. 따라서 출력 위너 어드레스는 이진수 111 또는 7이다. 노드(7)은 노드(5 및 7) 사이의 매칭을 이긴다. 플립 플롭(963)은 이 승리 결과(a0)로 업데이트된다.

제 3 매칭을 위하여, 부모 로직(303)은 노드(10), 노드(13 또는 이진수 1101)의 부모를 출력하고, 이 어드레스를 플립 플롭 어레이에 제공한다. 최상위 3개의 비트가 110이기 때문에, 선택 박스(1005)는 w2로서 n0의 보수를 선택한 2를 출력하고 선택 박스(1006)은 역시 2를 출력하고, w1으로서 멀티플렉서(1001)의 출력을 선택한다. w2가 제로값이기 때문에, 멀티플렉서(1001)은 플립플롭(962)의 상태, w1에 대하여 1을 선택한다. n3, w2 및 w1의 결합은 w00에 대한 플립 플롭(959)의 상태를 선택한다. 따라서 위너 어드레스는 리프 노드(2)인 0100이다. 따라서, 제 3 매칭은 노드 2 및 노드 7 사이이다. 노드(2)는 이 매칭을 이긴다. 플립 플롭(964)은 이 승리 결과로 업데이트되고 상기 토너먼트는 도 16에 도시된 상태로 업데이트된 플립 플롭 어레이가 완료된다.

멀티 스테이지 토너먼트 소트

많은 경우 도 17에 도시된 바와같이 포켓 스케줄 처리를 다중 스테이지로 분할하는 것이 바람직하다. 예를들어, 사람은 출력 트렁크와 연관된 대역폭을 몇몇 '부분' 트렁크로 분할하길 원한다. 패킷은 각각의 '부분' 트렁크에 대한 다음 몇몇 패킷을 선택하기 위하여 토너먼트 소트의 일세트를 우선 시행하고 부분 트렁크 큐에서 이를 패킷을 큐합으로써 스케줄된다. 그 다음, 제 2 토너먼트 소트는 라인상에 부분 트렁크 큐로부터 패킷을 크세를하기 위하여 수행된다. 일실시예로서, 섬유상에 전송된 데이타의 각각의 다수의 부분은 스케줄링 방법의 여러 조합에 따라 분류될수있다. 일부분은 가장 효율적인 방법, 양쪽 CBR 및 가장 우수한 방법에 따른 다른 방법 등에 따라 스케줄된다. 최종 스테이지에서, 모든 큐의 스케줄은 대역폭의 공유를 각각의 부분이 얻는 것을 보장하기 위하여 CBR 방법에 따를수있다.

제 2 실시예에서, 패킷 스케줄링 기능은 시스템의 두개의 다른 모듈 또는 칩상에서 분할된다. 이 경우, 하나의 세트의 토너먼트 소트는 몇몇 중간 큐 각각에 대한 다음 세트의 패킷을 선택하기 위하여 제 1 모듈에서 수행되고, 각각의 토너먼트 소트는 중간 큐의 헤드로부터 출력 라인에 패킷을 스케줄하기 위하여 제 2 모듈에서 수행된다.

당업자는 여기에 기술된 출력 패킷 스케줄러상에 많은 변형이 있다는 것을 이해할것이다. 예를들어 3개 이상의 스케줄링 방법은 여기에 기술된 두개(CBR 및 WFQ)의 결합일수있다. 다른 소팅 알고리즘은 전송될 다음 패킷을 선택하기 위하여 사용될수있다. 스케줄된 유니트는 IP 라우터 같은 패킷, ATM 스위치 같은 셀, 또는 프레임 릴레이 스위치 같은 프레임일수있다.

본 발명이 바람직한 실시예를 참조하여 도시되고 기술되었지만, 당업자는 첨부된 청구범위에 의해 한정된 본 발명의 사상으로부터 벗어나지 않고 형태 및 상세한 항목의 다수의 변화가 있을수있다는 것을 이해할 것이다.

(57) 청구의 범위

청구항 1

전송될 데이터 패킷을 저장하는 대기열; 및

패킷이 전송되는 대기열을 선택하는 스케줄러를 포함하며,

상기 스케줄러는:

상기 대기열과 관련된 스케줄링 값; 및

전송될 패킷을 선택하기 위해 스케줄링 값이 비교되는 선택 네트워크를 포함하는 네트워크 라우터.

청구항 2

제1항에 있어서,

상기 선택 네트워크는 트리 구조이고, 트리 구조의 리프(leaf)는 대기열의 스케줄링 값을 표현하며, 트리 구조의 내부 노드은 트리 구조의 현제 노드의 스케줄링 값의 비교에서 승자를 나타내는 네트워크 라우터.

40-10

청구항 3

제2항에 있어서,

상기 스케줄러는 스케줄링 값 비교를을 트리 구조를 통해 변화된 스케줄링 값을 나타내는 리프 노드로부터 트리 구조의 루트(root)로 경로를 제한하는 네트워크 라우터.

청구항 4

제2항에 있어서, 트리 구조의 내부 노드들은 승리한 형제 노드들로부터의 스케줄링 값을 저장하는 네트워크 라우터.

청구항 5

제4항에 있어서,

내부 노즈들은 저장된 스케줄링 값에 대응하는 리프 노드들의 신원을 저장하는 네트워크 라우터.

청구항 6

제2항에 있어서,

상기 스케줄러는 트리 구조를 저장하는 랜덤 엑세스 메모리(RAM), 비교될 스케줄링 값을 RAM 으로부터 접속하는 어드레스를 저장하는 어드레스 레지스터, RAM 으로부터의 스케줄링 값에 비교될 스케줄링 값을 저장하는 비교 레지스터 및 스케줄링 값들을 비교하는 비교기를 포함하는 네트워크 라우터.

청구항 7

제6항에 있어서,

상기 스케줄러는 어드레스 레지스터에서 어드레스를 수신하고 비교될 스케줄링 값이 저장된 형제 노드를 결정하며 승리한 비교 스케줄링 값이 저장된 부모 노드 어드레스를 결정하는 하드웨어를 추가로 포함하는 네트워크 라우터.

청구항 8

제2항에 있어서,

상기 스케줄러는 파이프라인 스테이지들을 포함하고, 파이프 라인 스테이지를 각각은 트리 구조의 개별 구분들에 의해 표시되는 스케줄링 값을 비교하는 네트워크 라우터.

청구항 9

제8항에 있어서,

상기 스케줄러는 상기 파이프라인 스테이지들을 가로질러 파티션화된 랜덤 엑세스 메모리를 포함하고, 각 파티션은 트리 구조의 적어도 한개 레벨을 저장하는 네트워크 라우터.

청구항 10

제9항에 있어서,

상기 스케줄러는 각 파이프라인 스테이지에서 RAM 으로부터 비교될 스케줄링 값에 접속하는 어드레스를 저장하는 어드레스 레지스터, 비교될 스케줄링 값을 RAM 으로부터 스케줄링 값에 저장하는 비교 레지스터, 및 상기 스케줄링 값들을 비교하는 비교기를 추가로 포함하는 네트워크 라우터.

청구항 11

제2항에 있어서,

각 노드는 승리 리프 노드에 대한 경로를 식별하는 네트워크 라우터.

청구항 12

제11항에 있어서,

리프 노드들을 저장하는 랜덤 엑세스 메로리, 각 내부 노드에서 승자를 식별하는 플립플롭 어레이 및 플립플롭 어레이에 저장된 데이타에 의해 표시되는 RAM으로부터의 리프 노드들의 스케줄링 값을 비교하는 비교기를 포함하는 네트워크 라우터.

청구항 13

제2항에 있어서,

각 대기열과 관련된 표시기를 추가로 포함하여 이러한 대기열이 스케줄링으로부터 디스에이블되도록 하는 네트워크 라우터.

청구항 14

제2항에 있어서,

상기 스케줄링 값들은 고정 비트율(CBR) 서비스 보증에 따라 스케줄된 전송 시간들을 포함하는 네트워크 라우터.

40-11

청구항 15

제14항에 있어서,

상기 스케줄링 값들은 가변 패킷 링크들을 반영하기 위해 업데이트되는 네트워크 라우터.

청구항 16

제14항에 있어서,

상기 스케줄링 값들은 이전 패킷에 인가된 바이트 스터핑을 반영하기 위해 업데이트되는 네트워크 라우터.

청구항 17

제14항에 있어서,

가중-공평-대기(WFQ) 스케줄링 방식을 사용하여 이론적인 전송 시간들을 나타내는 스케줄링 값들을 포함하는 네트워크 라우터.

청구항 18

제17항에 있어서,

상기 WFQ 스케줄링 값들은 가변 패킷 길이들을 위해 업데이트되는 네트워크 라우터.

청구항 19

제17항에 있어서,

상기 스케줄링 값들은 이전 패킷에 인가되는 바이트 스터핑을 반영하기 위해 업데이트되는 네트워크 라우터.

청구항 20

제2항에 있어서,

가중-공평-대기(WFQ) 스케줄링 방식을 사용하여 이론적인 전송 시간들을 나타내는 스케줄링 값들을 추가로 포함하는 네트워크 라우터.

청구항 21

제20항에 있어서,

상기 WFQ 스케줄링 값들은 가변 패킷 길이를 위해 업데이트되는 네트워크 라우터.

청구항 22

제20항에 있어서,

상기 스케줄링 값들은 이전 패킷에 인가되는 바이트 스터핑을 반영하기 위해 업데이트되는 네트워크 라우터.

청구항 23

제1항에 있어서,

상기 선택 네트워크는 네트워크를 분류(sorting)하여 스케줄링 우선성에 의해 대기열들을 순서화하도록 스케줄링 값들이 비교되는 네트워크 라우터.

청구항 24

전송될 데이터 패킷을 저장하는 대기열; 및

패킷들이 전송될 대기열을 선택하는 스케줄러를 포함하며,

상기 스케줄러는

대기열들과 관련된 스케줄링 값들;

대기열들을 디스에이블하기 위해 대기열들과 관련된 표시기; 및

데이터 패킷을 전송하기 위해 디스에이블 되지 않는 대기열들의 스케줄링 값들을 비교하는 비교기를 포함하는 네트워크 라우터.

청구항 25

전송될 데이터 패킷을 저장하는 대기열; 및

패킷들이 전송될 대기열을 선택하는 스케줄러를 포함하며,

상기 스케줄러는

제1 대기열 서브셋과 관련된 제1 스케줄링 방법에 상응하는 제1 스케줄링 값들;

40-12

제2 대기열 서브셋과 관련된 제2 스케쥴링 방법에 상응하는 제2 스케쥴링 값들로서, 적어도 하나의 대기열은 대기열의 제1 서브셋 및 제2 서브셋 중 각각의 멤버인 제2 스케쥴링 값들; 및

제1 스케쥴링 값들 및 제2 스케쥴링 값들이 비교되어 전송될 패킷들을 선택하는 대기열 선택기를 포함하는 네트워크 라우터.

청구항 26

제25항에 있어서,

상기 제1 스케쥴링 방법은 고정 비트율(CBR) 스케쥴링 이고 상기 제2 스케쥴링 방법은 가중-공평-대기(WFQ) 스케쥴링인 네트워크 라우터.

청구항 27

제26항에 있어서,

상기 스케쥴러는

가장 빨리 스케쥴된 CBR 대기열을 식별하고;

식별된 CBR 대기열의 스케쥴링 값이 현재 시간 이하인 경우, CBR 대기열로부터 대응 패킷을 전송하고 대기열과 관련된 CBR 스케쥴링 값을 업데이트 하며; 그리고

식별된 CBR 대기열의 스케쥴링 값이 현재 시간을 초과하는 경우, 가장 빠른 스케쥴링 값을 갖는 WFQ 대기열로부터 패킷을 전송하고 그 대기열의 스케쥴링 값을 업데이트함으로써 대기열을 선택하는 네트워크 라우터.

청구항 28

전송될 데이터 패킷들을 저장하는 대기열; 및

패킷들이 전송되는 대기열을 선택하는 스케쥴러를 포함하며,

상기 스케쥴러는

대기열과 관련된 스케쥴링 값들;

전송될 패킷들을 선택하기 위해 스케쥴링 값들이 비교되는 선택기; 및

대기열에서 패킷의 가변 길이에 따라 대기열의 스케쥴링 값을 업데이트하는 스케쥴링 값 업데이터를 포함하는 네트워크 라우터.

청구항 29

제28항에 있어서,

상기 스케쥴링 값들은 이전 패킷에 인가된 바이트 스터핑을 반영하기 위해 업데이트되는 네트워크 라우터.

청구항 30

제29항에 있어서,

상기 스케쥴러는

가장 빨리 스케쥴된 CBR 대기열을 식별하고;

식별된 CBR 대기열의 스케쥴링 값이 현재 시간 이하인 경우, CBR 대기열로부터 대응 패킷을 전송하고 대기열과 관련된 CBR 스케쥴링 값을 업데이트 하며; 그리고

식별된 CBR 대기열의 스케쥴링 값이 현재 시간을 초과하는 경우, 가장 빠른 스케쥴링 값을 갖는 WFQ 패킷으로부터 패킷을 전송하고 그 대기열의 스케쥴링 값을 업데이트함으로써 대기열을 선택하는 네트워크 라우터.

청구항 31

제28항에 있어서,

상기 스케쥴러는

가장 빨리 스케쥴된 CBR 대기열을 식별하고;

식별된 CBR 대기열의 스케쥴링 값이 현재 시간 이하인 경우, CBR 대기열로부터 대응 패킷을 전송하고 대기열과 관련된 CBR 스케쥴링 값을 업데이트 하며; 그리고

식별된 CBR 대기열의 스케쥴링 값이 현재 시간을 초과하는 경우, 가장 빠른 스케쥴링 값을 갖는 WFQ 패킷으로부터 패킷을 전송하고 그 대기열의 스케쥴링 값을 업데이트함으로써 대기열을 선택하는 네트워크 라우터.

청구항 32

전송될 데이터 패킷들을 저장하는 제1 대기열 셋;

40-13

제1 중간 대기열로 패킷들이 전송되는 제1 대기열 셋의 대기열들을 선택하는 제1 스케줄러;

전송될 데이타 패킷들을 저장하는 제2 대기열 셋;

제2 중간 대기열로 패킷들이 전송되는 제2 대기열 셋의 대기열들을 선택하는 제1 스케줄러;

패킷들이 전송되는 중간 대기열들을 선택하는 추가적인 스케줄러를 포함하는 네트워크 라우터.

청구항 33

제32항에 있어서,

상기 제1 스케줄러는 복수 스케줄링 방법들에 따라 대기열들을 선택하는 네트워크 라우터.

청구항 34

패킷들을 라우팅하는 방법으로서,

데이터 패킷들을 대기열들에 저장하는 단계;

스케줄링 값들을 대기열들과 관련시키는 단계; 및

선택 네트워크에서 스케줄링 값들을 비교하여 패킷들이 전송되는 대기열들을 선택하는 단계를 포함하는 데이터 패킷 라우팅 방법.

청구항 35

제34항에 있어서,

상기 선택 네트워크는 트리 구조이고, 트리 구조의 리프(leaf)는 대기열의 스케줄링 값을 표현하며, 트리 구조의 내부 노드들은 트리 구조의 형제 노드의 스케줄링 값들의 비교에서 승자를 나타내는 데이터 패킷 라우팅 방법.

청구항 36

제35항에 있어서,

상기 스케줄러는 스케줄링 값 비교들을 트리 구조를 통해 변화된 스케줄링 값을 나타내는 리프 노드로부터 트리 구조의 루트(root)로 경로를 제한하는 데이터 패킷 라우팅 방법.

청구항 37

제35항에 있어서,

트리 구조의 내부 노드들은 승리한 형제 노드들로부터의 스케줄링 값을 저장하는 데이터 패킷 라우팅 방법.

청구항 38

제37항에 있어서,

내부 노드들은 저장된 스케줄링 값에 대응하는 리프 노드들의 신원을 저장하는 데이터 패킷 라우팅 방법.

청구항 39

제35항에 있어서,

상기 트리 구조는 랜덤 엑세스 메모리(RAM)에 저장되고, 비교 레지스터 및 RAM으로부터의 스케줄링 값들이 비교되는 데이터 패킷 라우팅 방법.

청구항 40

제39항에 있어서,

비교될 스케줄링 값이 저장되는 현재 노드를 결정하는 단계 및 승리한 비교 스케줄링 값이 저장되는 부모 노드 어드레스를 결정하는 단계로 추가로 포함하는 데이터 패킷 라우팅 방법.

청구항 41

제35항에 있어서,

파이프라인 스테이지들의 트리 구조의 개별 부분들에 의해 표시되는 스케줄링 값들을 비교하는 단계를 추가로 포함하는 데이터 패킷 라우팅 방법.

청구항 42

제41항에 있어서,

파이프라인 스테이지들을 가로질러 파티션화된 랜덤 엑세스 메모리(RAM)의 파티션에서 트리 구조의 적어도 한개 레벨을 저장하는 단계를 추가로 포함하는 데이터 패킷 라우팅 방법.

청구항 43

제42항에 있어서,

40-14

각 파이프라인 스테이지에서, 비교 레지스터 및 RAM 으로부터의 스케쥴링 값들을 비교하는 단계를 추가로 포함하는 데이터 패킷 라우팅 방법.

청구항 44

제35항에 있어서,

각 노드는 승리 리프 노드에 대한 경로를 식별하는 데이터 패킷 라우팅 방법.

청구항 45

제44항에 있어서,

트리 구조의 리프 노드들은 랜덤 엑세스 메모리에 저장되고, 각 내부 노드의 승자는 플립플롭 어레이에서 식별되며, 상기 방법은 플립플롭 어레이에 저장된 데이터에 의해 표시되는 RAM 으로부터의 리프 노드들의 스케쥴링 값들을 비교하는 단계를 포함하는 데이터 패킷 라우팅 방법.

청구항 46

제34항에 있어서,

스케쥴링으로부터 대기열을 디스에이블하기 위해 각 대기열과 관련된 표시기를 제공하는 단계를 추가로 포함하는 데이터 패킷 라우팅 방법.

청구항 47

제34항에 있어서,

고정 비트율(CBR) 서비스 보증에 따라 스케쥴된 전송 시간들을 포함하는 데이터 패킷 라우팅 방법.

청구항 48

제47항에 있어서,

상기 스케쥴링 값들은 가변 패킷 링크들을 반영하기 위해 업데이트되는 데이터 패킷 라우팅 방법.

청구항 49

제47항에 있어서,

상기 스케쥴링 값들은 이전 패킷에 인가된 바이트 스터핑을 반영하기 위해서 업데이트되는 데이터 패킷 라우팅 방법.

청구항 50

제47항에 있어서,

스케쥴링 값들은 가중-공평-대기(WFQ) 스케쥴링 방식을 사용하여 이론적인 전송 시간들을 나타내는 데이터 패킷 라우팅 방법.

청구항 51

제50항에 있어서,

상기 WFQ 스케쥴링 값들은 가변 패킷 길이들을 위해 업데이트되는 데이터 패킷 라우팅 방법.

청구항 52

제50항에 있어서,

상기 스케쥴링 값들은 이전 패킷에 인가된 바이트 스터핑을 반영하기 위해 업데이트되는 데이터 패킷 라우팅 방법.

청구항 53

제34항에 있어서,

스케쥴링 값들은 가중-공평-대기(WFQ) 스케쥴링 방식을 사용하여 이론적인 전송 시간들을 표현하는 데이터 패킷 라우팅 방법.

청구항 54

제53항에 있어서,

상기 WFQ 스케쥴링 값들은 가변 패킷 길이들을 위해 업데이트되는 데이터 패킷 라우팅 방법.

청구항 55

제53항에 있어서,

상기 스케쥴링 값들은 이전 패킷에 인가된 바이트 스터핑을 반영하기 위해서 업데이트되는 데이터 패킷 라우팅 방법.

청구항 56

40-15

제34항에 있어서,

상기 선택 네트워크는 네트워크를 분류하며, 이를 통해 스케줄링 값들이 비교되어 스케줄링 우선성에 의해 대기열들을 순서화하도록 하는 데이터 패킷 라우팅 방법.

청구항 57

대기열에 데이터 패킷들을 저장하는 단계;

스케줄링 값들을 대기열들과 관련시키는 단계;

대기열들을 디스에이블시키기 위해 표시기르을 대기열들과 관련시키는 단계; 및

데이타 패킷들 전에 디스에이블되는 대기열들의 스케줄링 값들을 비교하는 단계로 포함하는 데이터 패킷 라우팅 방법.

청구항 58

대기열에 데이타 패킷들을 저장하는 단계;

제1 스케줄링 방법에 대응하는 스케줄링 값들을 제1 대기열 서브셋과 관련시키는 단계;

제2 스케줄링 방법에 대응하는 스케줄링 값들을 제2 대기열 서브셋에 관련시키는 단계로서, 적어도 하나의 대기열은 상기 제1 및 제2 대기열 서브셋 각각의 멤버인 단계; 및

전송될 패킷들을 선택하기 위해 스케줄링 값들을 비교하는 단계를 포함하며, 제1 스케줄링 방법 하에서의 초과 용량은 제2 스케줄링 방법 하에서의 스케줄링을 위해 제공되는 데이터 패킷 라우팅 방법.

청구항 59

제58항에 있어서,

상기 제1 스케줄링 방법은 고정 비트율(CBR) 스케줄링이고 제2 스케줄링 방법은 가중-공평-대기(WFQ) 스케줄링인 데이터 패킷 라우팅 방법.

청구항 60

제58항에 있어서,

상기 스케줄러는

가장 빨리 스케줄된 CBR 대기열을 식별하고;

식별된 CBR 대기열의 스케줄링 값이 현재 시간 이하인 경우, CBR 대기열로부터 대응 패킷을 전송하고 대기열과 관련된 CBR 스케줄링 값을 업데이트 하며; 그리고

식별된 CBR 대기열의 스케줄링 값이 현재 시간을 초과하는 경우, 가장 빠른 스케줄링 값을 갖는 WFQ 대기열로부터 패킷을 전송하고 그 대기열의 스케줄링 값을 업데이트함으로써 대기열을 선택하는 데이터 패킷 라우팅 방법.

청구항 61

대기열에 데이터 패킷들을 저장하는 단계;

스케줄링 값들을 대기열과 관련시키는 단계;

전송될 데이타 패킷들을 선택하기 위해 스케줄 값들을 비교하는 단계; 및

대기열에서 가변 길이 패킷에 따라 대기열의 스케줄링 값을 업데이트하는 단계를 포함하는 데이터 패킷 라우팅 방법.

청구항 62

제61항에 있어서,

상기 스케줄링 값들은 이전 패킷에 인간된 바이트 스터핑을 반영하기 위해 업데이트되는 데이터 패킷 라우팅 방법.

청구항 63

제62항에 있어서,

상기 스케줄러는

가장 빨리 스케줄된 CBR 대기열을 식별하고;

식별된 CBR 대기열의 스케줄링 값이 현재 시간 이하인 경우, CBR 대기열로부터 대응 패킷을 전송하고 대기열과 관련된 CBR 스케줄링 값을 업데이트 하며; 그리고

식별된 CBR 대기열의 스케줄링 값이 현재 시간을 초과하는 경우, 가장 빠른 스케줄링 값을 갖는 WFQ 패킷으로부터 패킷을 전송하고 그 대기열의 스케줄링 값을 업데이트함으로써 대기열을 선택하는 데이터 패킷 라우팅 방법.

청구항 64

40-16

제61항에 있어서,

상기 스케줄러는

가장 빨리 스케줄된 CBR 대기열을 식별하고;

식별된 CBR 대기열의 스케줄링 값이 현재 시간 이하인 경우, CBR 대기열로부터 대응 패킷을 전송하고 대기열과 관련된 CBR 스케줄링 값을 업데이트 하며; 그리고

식별된 CBR 대기열의 스케줄링 값이 현재 시간을 초과하는 경우, 가장 빠른 스케줄링 값을 갖는 WFQ 패킷으로부터 패킷을 전송하고 그 대기열의 스케줄링 값을 업데이트하는 것을 포함하는 데이터 패킷 라우팅 방법.

청구항 65

제1 및 제2 대기열 셋에서 전송될 데이터 패킷을 저장하는 단계;

제1 중간 대기열로 전송되는 제1 대기열 셋의 대기열들을 선택하는 단계;

제2 중간 대기열로 전송되는 제2 대기열 셋의 대기열들을 선택하는 단계; 및

패킷들이 전송되는 중간 대기열들을 선택하는 단계를 포함하는 가장 빨리 스케줄된 CBR 대기열을 식별하고;

식별된 CBR 대기열의 스케줄링 값이 현재 시간 이하인 경우, CBR 대기열로부터 대응 패킷을 전송하고 대기열과 관련된 CBR 스케줄링 값을 업데이트 하며; 그리고

식별된 CBR 대기열의 스케줄링 값이 현재 시간을 초과하는 경우, 가장 빠른 스케줄링 값을 갖는 WFQ 패킷으로부터 패킷을 전송하고 그 대기열의 스케줄링 값을 업데이트함으로써 대기열을 선택하는 데이터 패킷 라우팅 방법.

청구항 66

제 65항에 있어서,

제1 대기열 셋의 대기열을 선택하는 상기 단계는 복수 스케줄링 방법에 따라 대기열들을 선택하는 단계를 추가로 포함하는 데이터 패킷 라우팅 방법.

청구항 67

전송될 데이타 패킷들을 저장하는 대기열; 및

패킷들이 전송되는 대기열들을 선택하는 스케줄링 수단을 포함하며,

상기 스케줄링 수단은

대기열과 관련된 스케줄링 값; 및

스케줄링 값들이 비교되어 전송될 패킷들을 선택하는 선택 네트워크를 포함하는 네트워크 라우터.

청구항 68

전송될 데이터 패킷들을 저장하는 대기열; 및

패킷들이 전송되는 대기열들을 선택하는 스케줄링 수단을 포함하며,

상기 스케줄링 수단은

대기열과 관련된 스케줄링 값;

대기열을 디스에이블시키는 대기열들과 관련된 표시 수단; 및

데이터 패킷들을 전송하기 위해 디스에이블되는 대기열의 스케줄링 값들을 비교하는 비교기 수단을 포함하는 네트워크 라우터.

청구항 69

전송될 데이터 패킷들을 저장하는 대기열; 및

패킷들이 전송되는 대기열들을 선택하는 스케줄링 수단을 포함하며,

상기 스케줄링 수단은

대기열 제1 서브셋과 관련된 제1 스케줄링 방법에 대응하는 제1 스케줄링 값;

대기열의 제2 서브셋과 관련되는 제2 스케줄링 방법에 대응하는 제2 스케줄링 값로서, 적어도 하나의 대기열의 제1 및 제2 대기열 서브셋 각각의 멤버인 제2 스케줄링 값; 및

제1 스케줄링 값들 및 제2 스케줄링 값들을 비교하여 전송될 패킷들을 선택하는 대기열 선택기 수단을 포함하는 네트워크 라우터.

청구항 70

전송될 데이터 패킷들을 저장하는 대기열; 및

40-17

패킷들이 전송되는 대기열들을 선택하는 스케줄링 수단을 포함하며,

상기 스케줄링 수단은

대기열들과 관련된 스케줄링 값들;

스케줄링 값들을 비교하여 전송될 패킷들을 선택하는 선택 수단; 및

대기열에서 가변 패킷 길이에 기초하여 대기열의 스케줄링 값을 업데이트하는 업데이트 수단을 포함하는 네트워크 라우터.

청구항 71

전송될 데이터 패킷들을 저장하는 제1 대기열 셋;

패킷이 제1 중간 대기열로 전송되는 제1 대기열 셋의 대기열들을 선택하는 제1 스케줄링 수단;

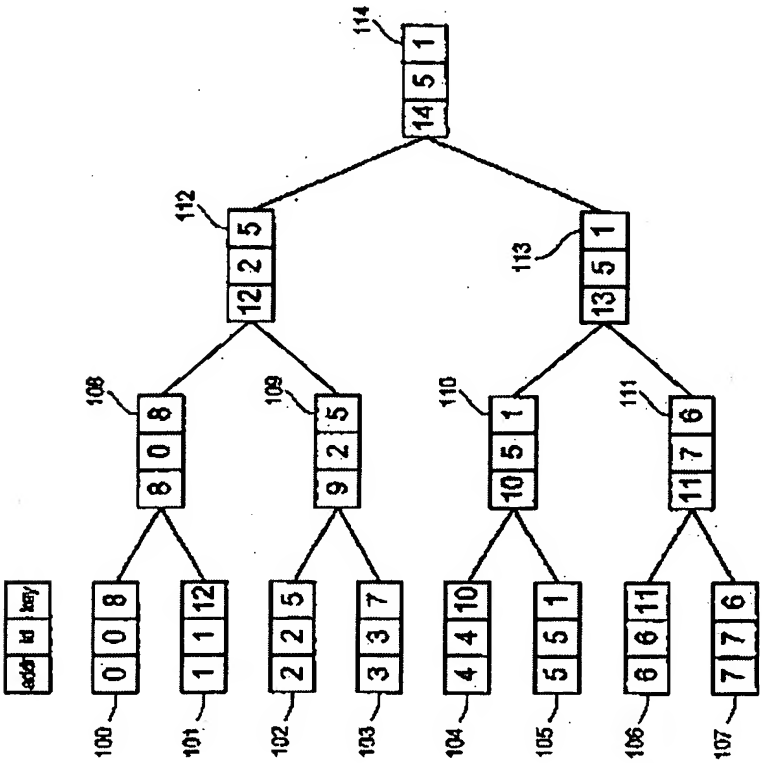
전송될 데이터 패킷들을 저장하는 제2 대기열 셋;

패킷이 제2 중간 대기열로 전송되는 제2 대기열 셋의 대기열들을 선택하는 제2 스케줄링 수단; 및

패킷들이 전송되는 중간 대기열을 선택하는 추가적인 스케줄링 수단을 포함하는 네트워크 라우터.

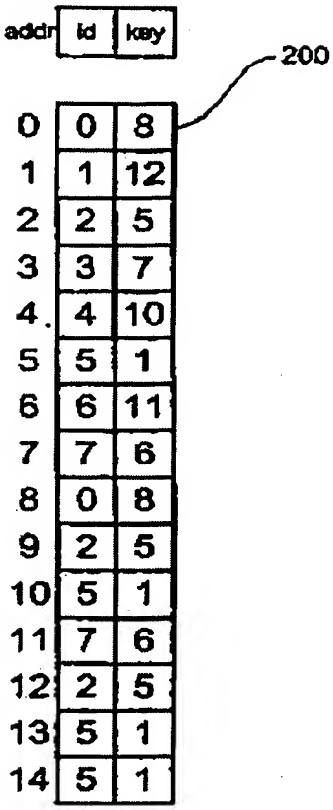
도면

도면1



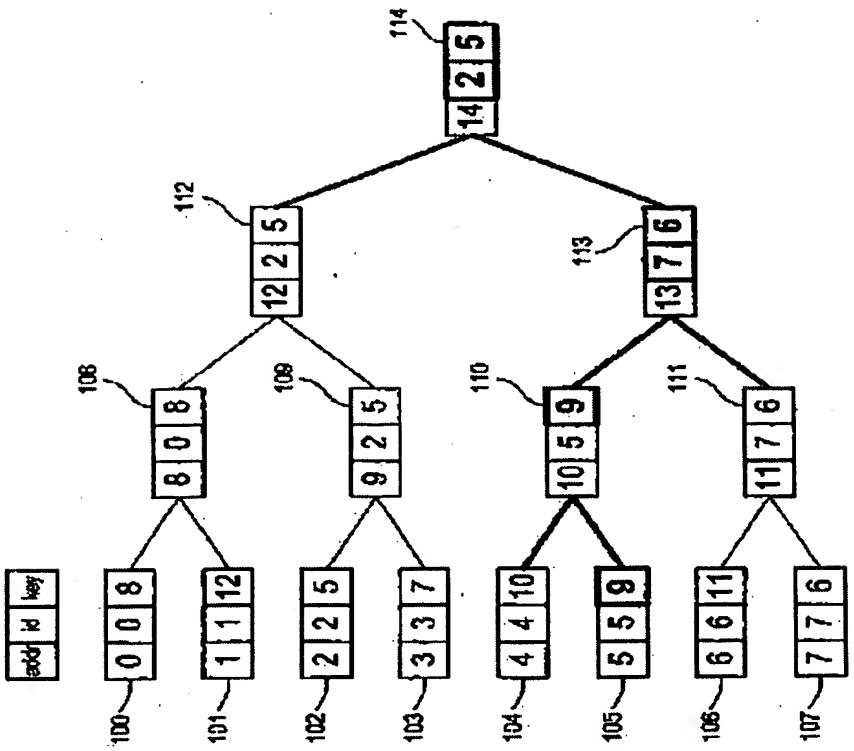
40-18

도면2


addr
id
key
200
0 0 8
1 1 12
2 2 5
3 3 7
4 4 10
5 5 1
6 6 11
7 7 6
8 0 8
9 2 5
10 5 1
11 7 6
12 2 5
13 5 1
14 5 1

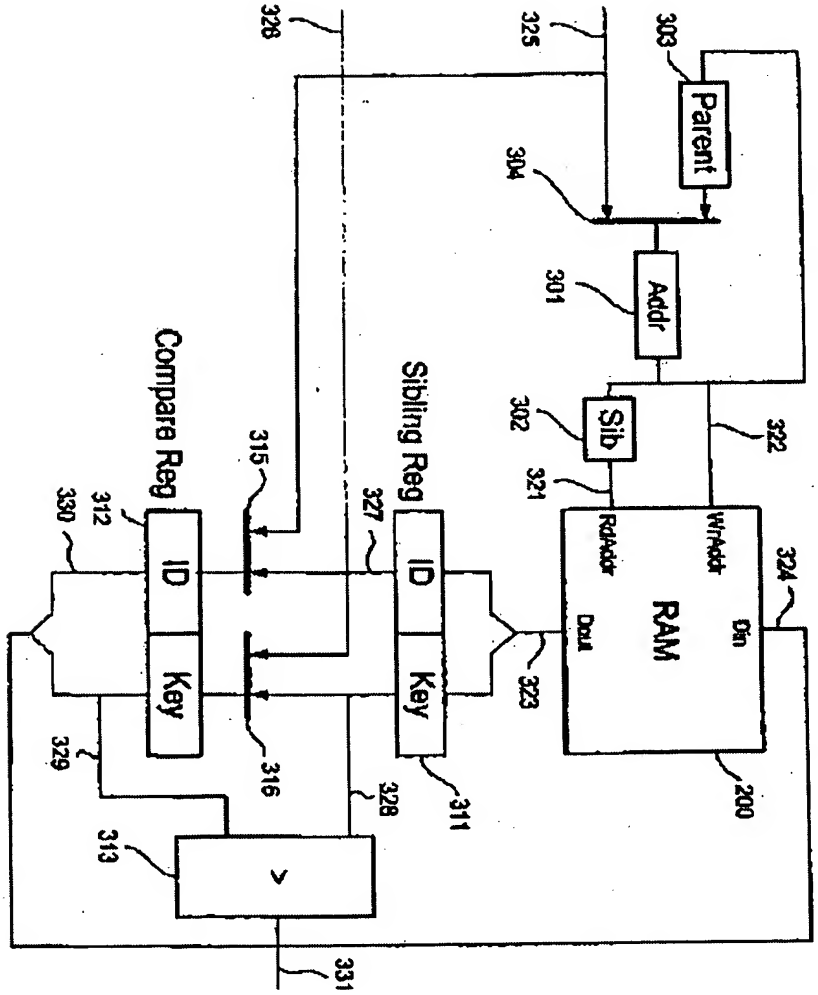
40-19

도면3


114
14 2 5
112
12 2 5
113
13 7 6
108
8 0 8
109
9 2 5
110
10 5 9
111
11 7 6
pointer id key
0 0 8
1 1 12
2 2 5
3 3 7
4 4 10
5 5 9
6 6 11
7 7 6
100
101
102
103
104
105
106
107

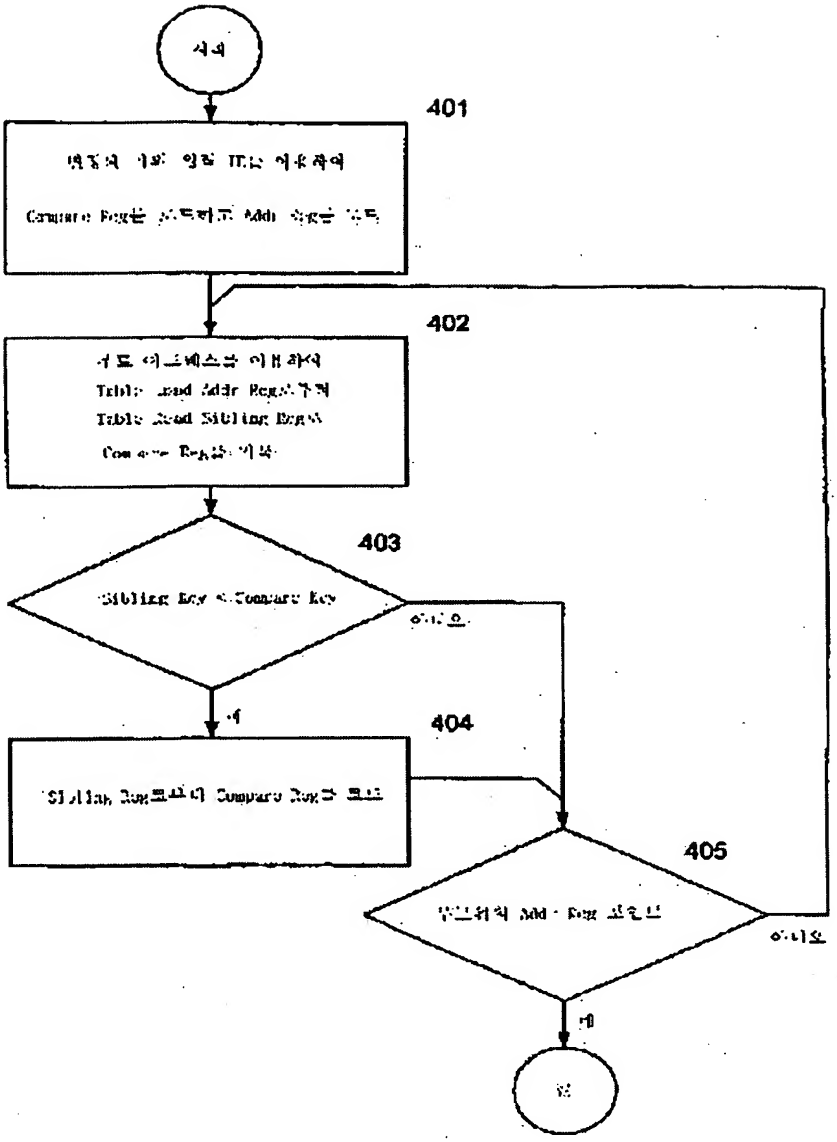
40-20

도면4


328
325
303
Parent
304
301
Addr
Compare Reg
315
Sibling Reg
302
Sib
322
321
330
312
327
RdAddr
WrAddr
324
ID
ID
RAM
Din
Dout
Key
Key
323
329
316
328
311
200
313
>
331

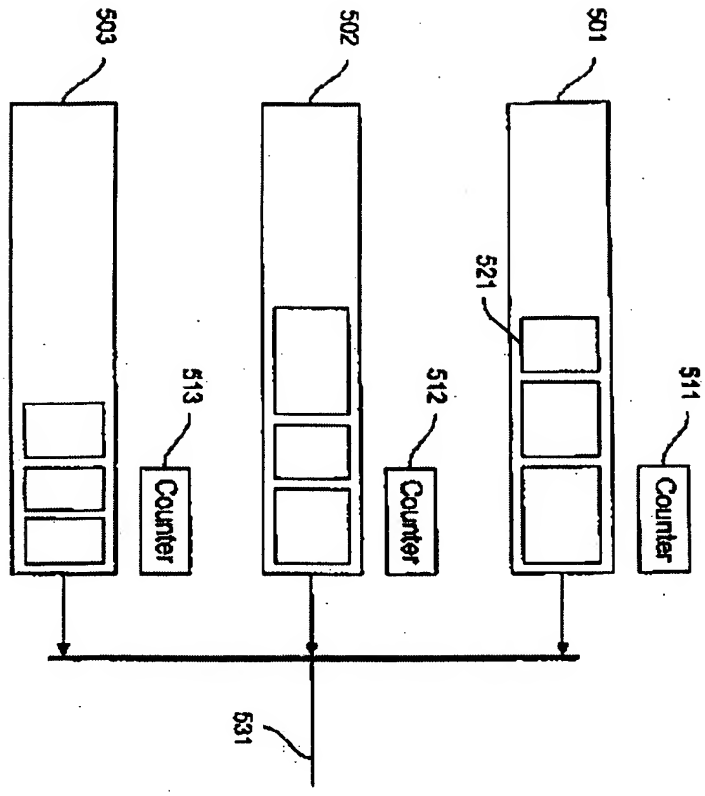
40-21

도면5



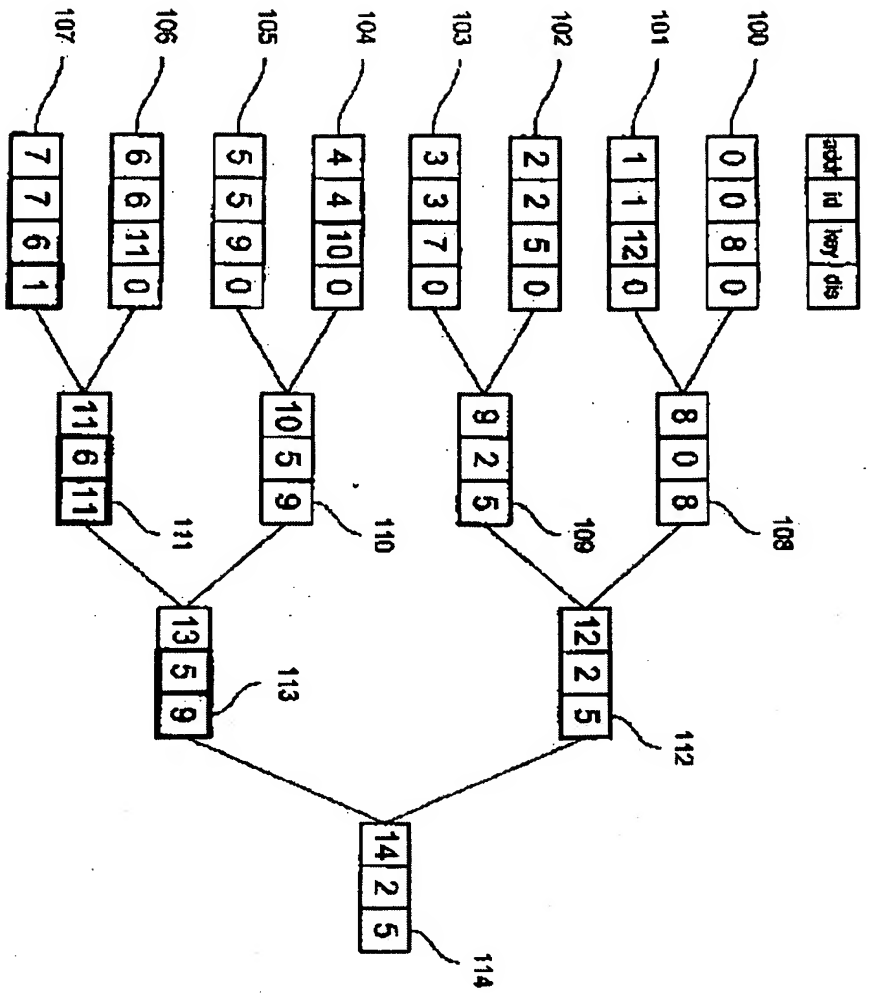
40-22

도면6



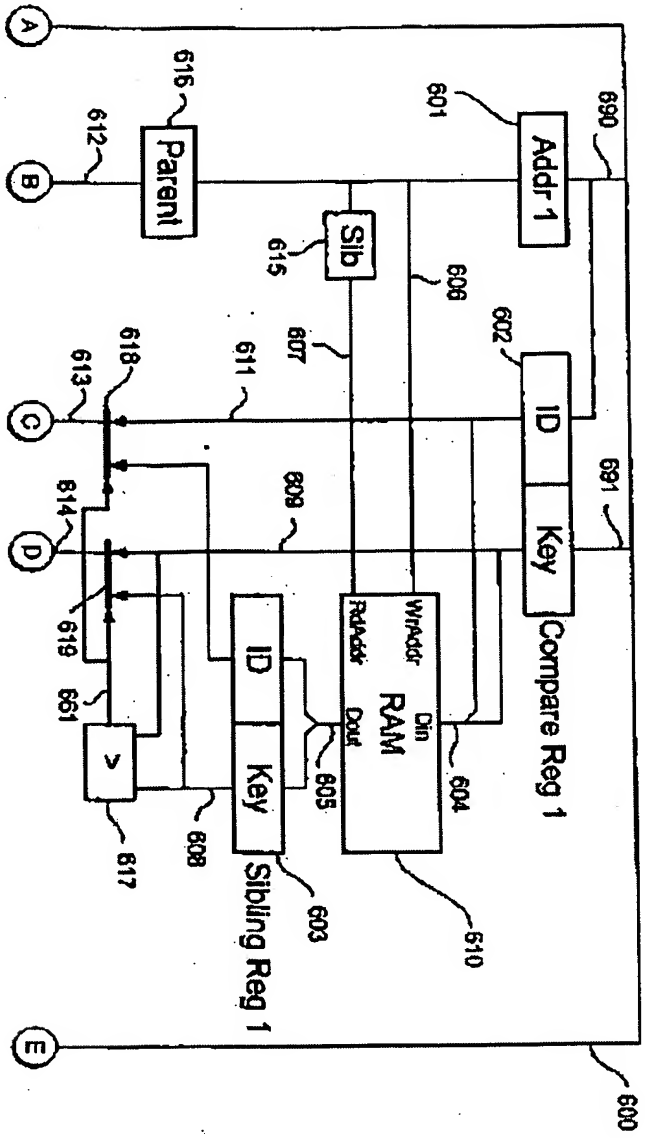
40-23

도면7


100
101
102
103
104
105
106
107
packet id key dis
0 0 8 0
1 1 12 0
2 2 5 0
3 3 7 0
4 4 10 0
5 5 9 0
6 6 11 0
7 7 6 1
8 0 8
9 2 5
10 5 9
11 6 11
12 2 5
13 5 9
14 2 5
108
109
110
111
112
113
114

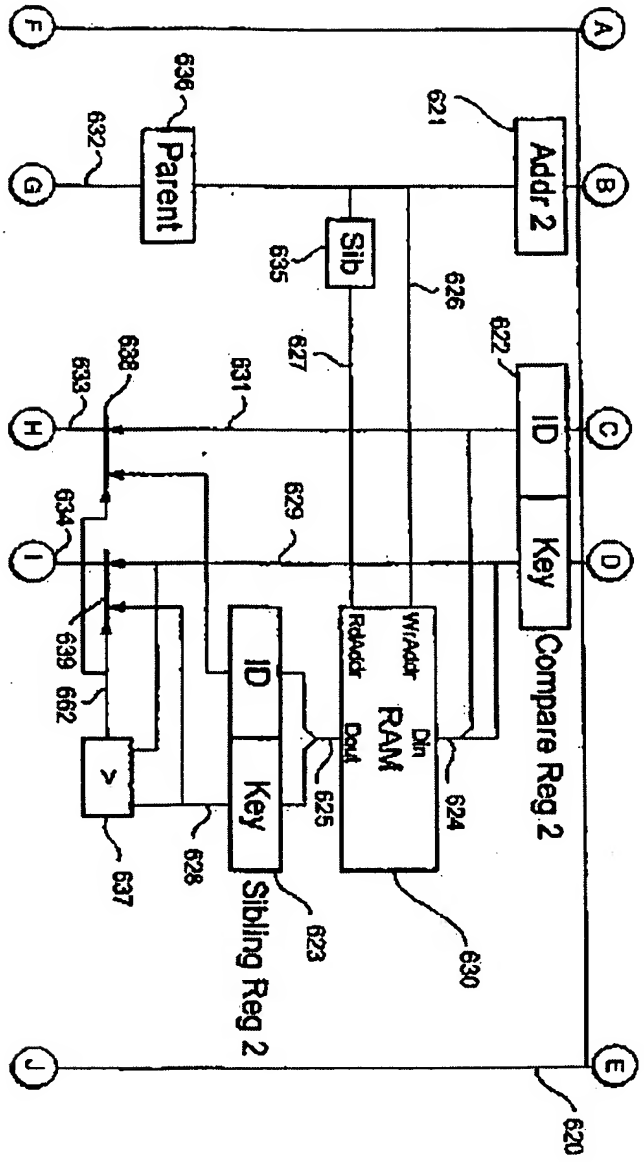
40-24

도면4


A
B
C
D
E
Parent
Addr 1
Slb
ID
Key
Compare Reg 1
ID
Key
Sibling Reg 1
RAM
RdAddr
WrAddr
Din
Dout
>
616
612
601
690
615
606
607
602
618
613
611
691
614
609
619
661
605
604
608
617
603
610
600

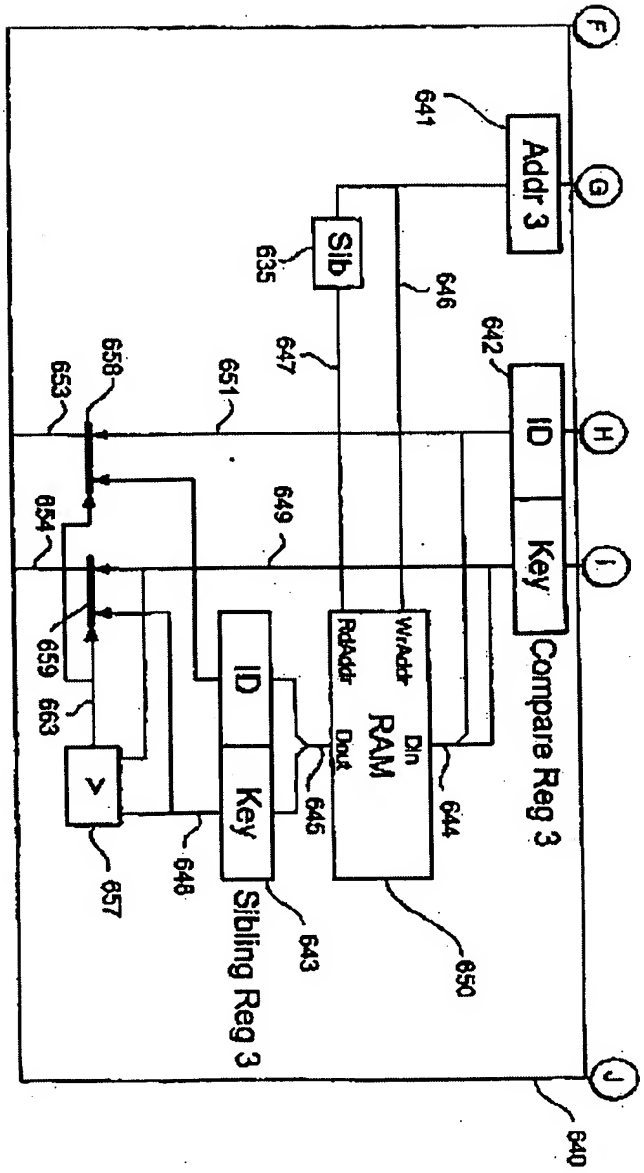
40-25

도면88


F
A
636
621
632
Parent
Addr 2
G
B
Sib
635
626
627
622
638
633
631
ID
H
C
634
629
Key
I
D
639
662
RdAddr
WrAddr
ID
Din
RAM
Dout
Compare Reg 2
625
624
>
Key
628
637
623
630
Sibling Reg 2
J
E
620

40-26

도면6c


F
641
Addr 3
G
635
Slb
646
642
647
653
658
651
ID
H
649
654
Key
I
659
663
WrAddr
RdAddr
Din
Dout
RAM
Compare Reg 3
ID
Key
645
644
646
657
643
650
Sibling Reg 3
J
640

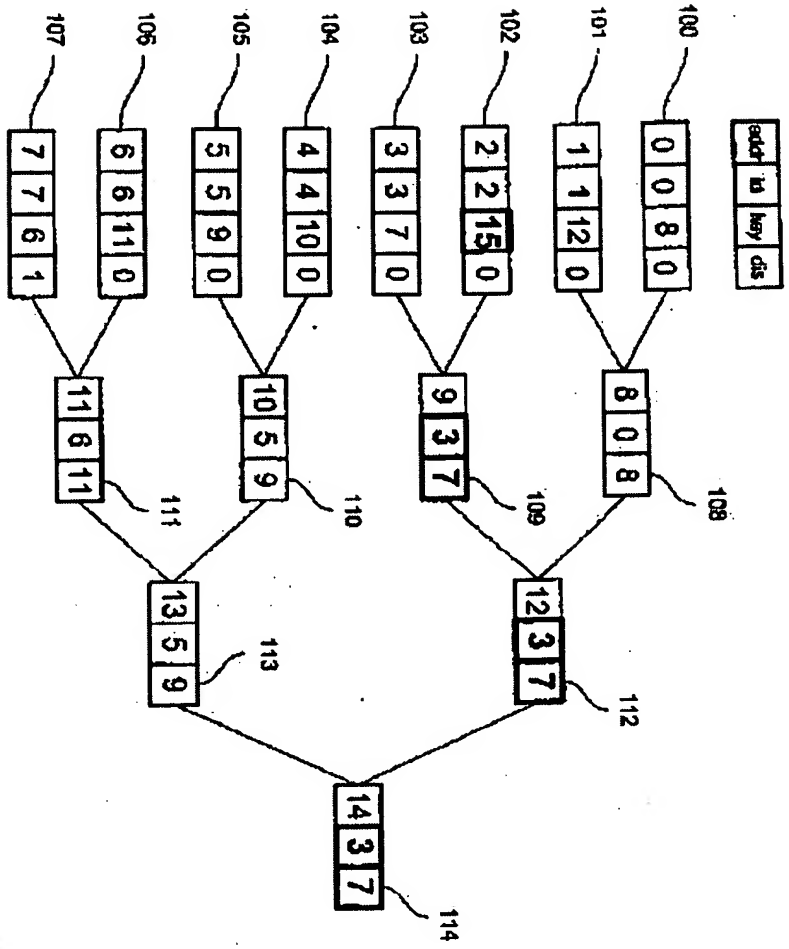
40-27

도면9

Register	Cycle				
	1	2	3	4	5
Addr 1	5	7	2		
Comp 1	5.9	7.d	2.15		
Sib 1	4.10	6.11	3.7		
Addr 2		10	11	9	
Comp 2		5.9	6.11	3.7	
Sib 2		7.6	5.9	0.8	
Addr 3			13	13	12
Comp 3			7.6	5.9	3.7
Sib 3			2.5	2.5	5.9
Result			2.5	2.5	3.7

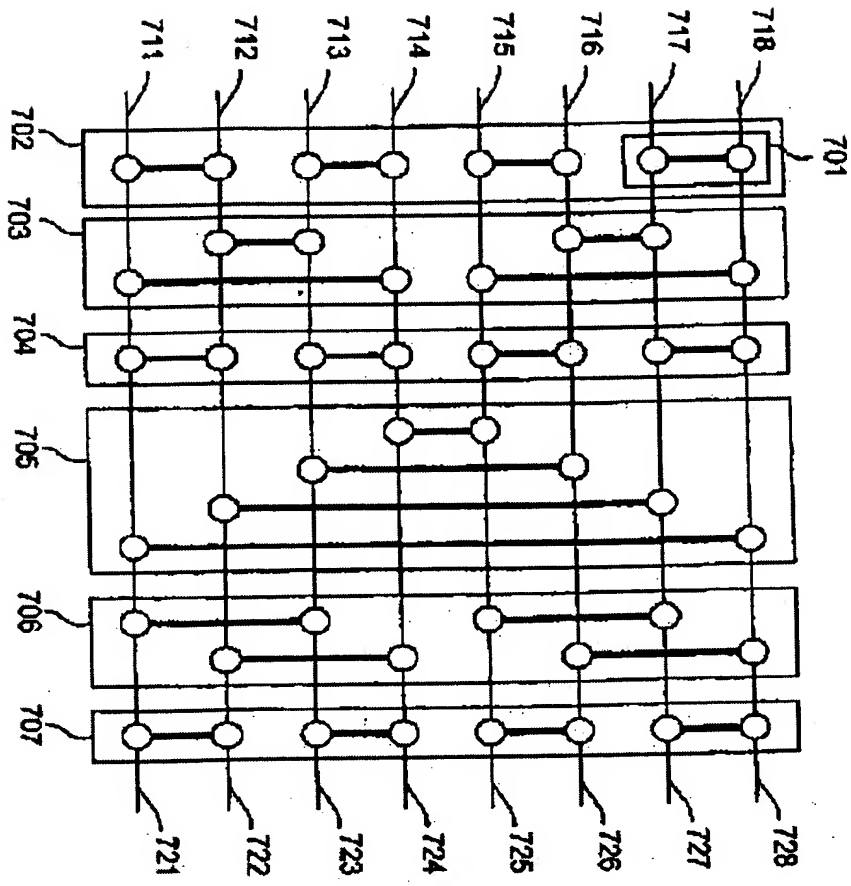
40-28

도면10


addr
id
key
dis
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

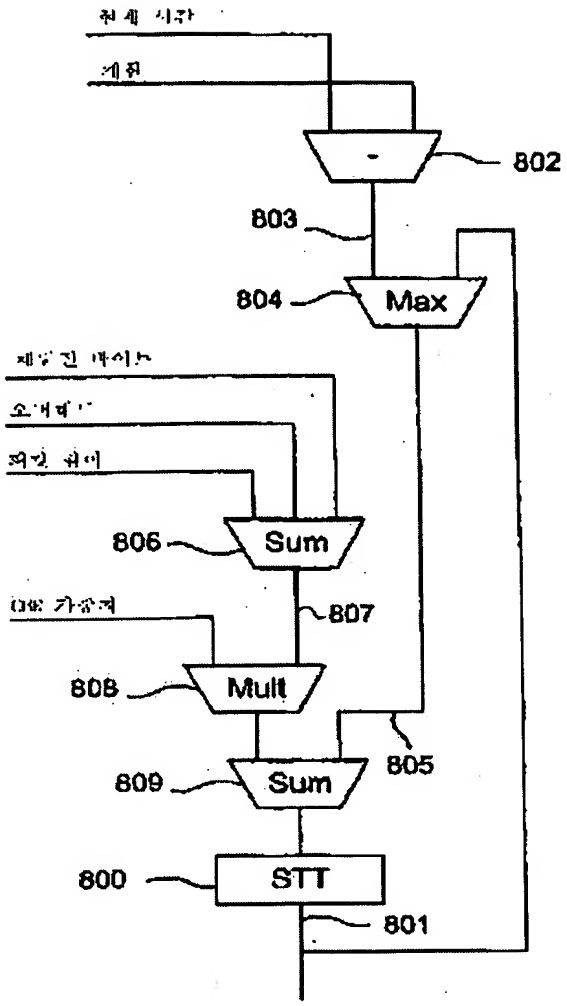
40-29

도면11


711
712
713
714
715
716
717
718
701
702
703
704
705
706
707
721
722
723
724
725
726
727
728

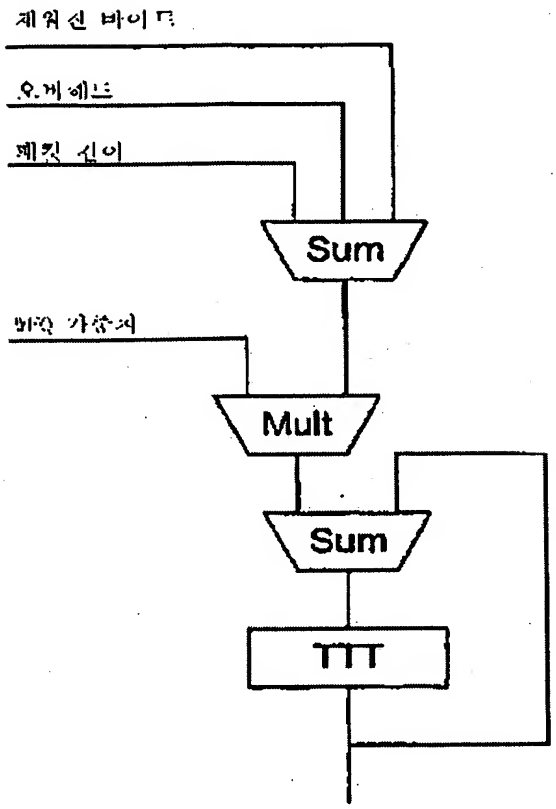
40-30

도면12



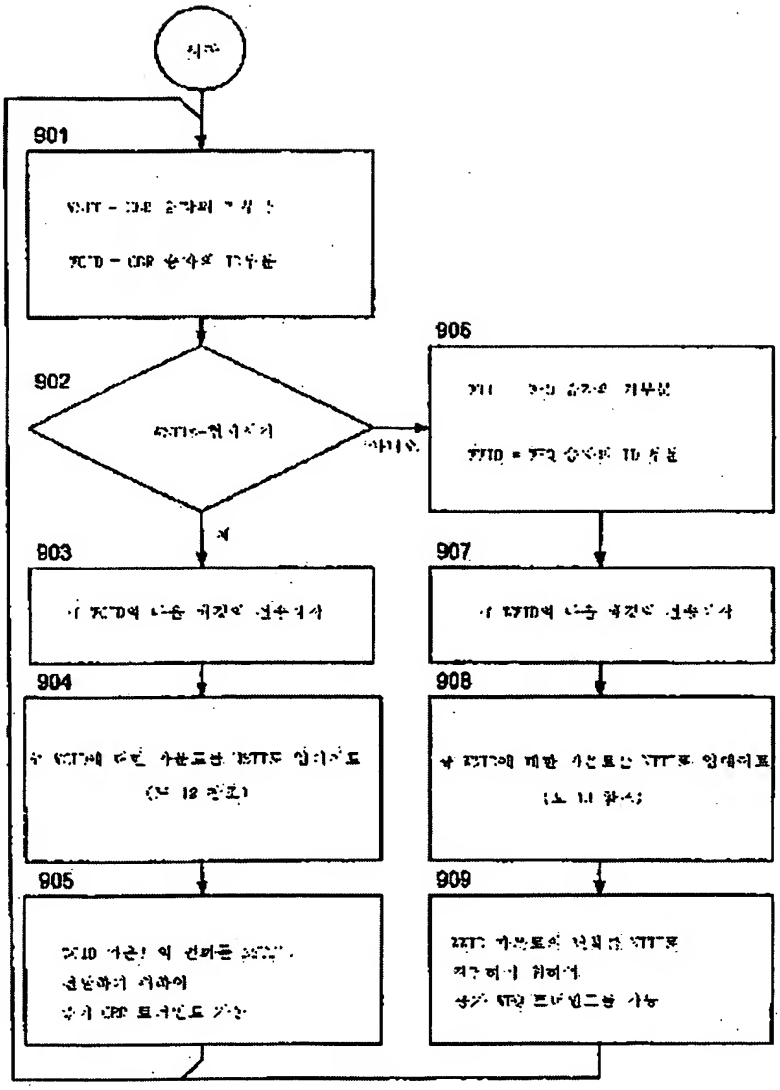
40-31

도면13



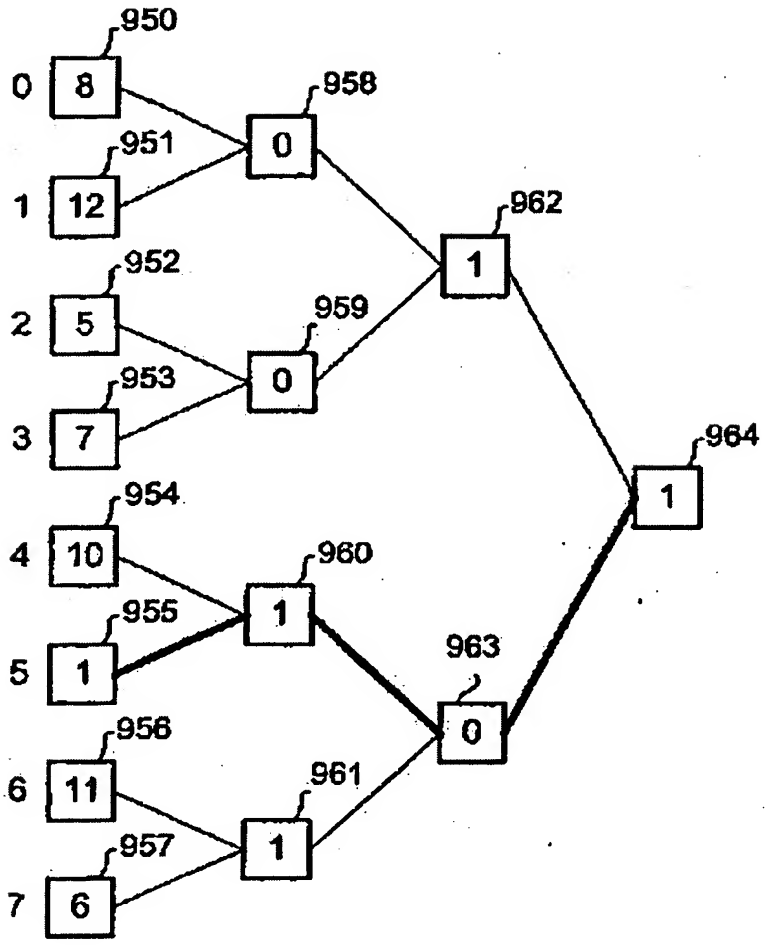
40-32

도면14



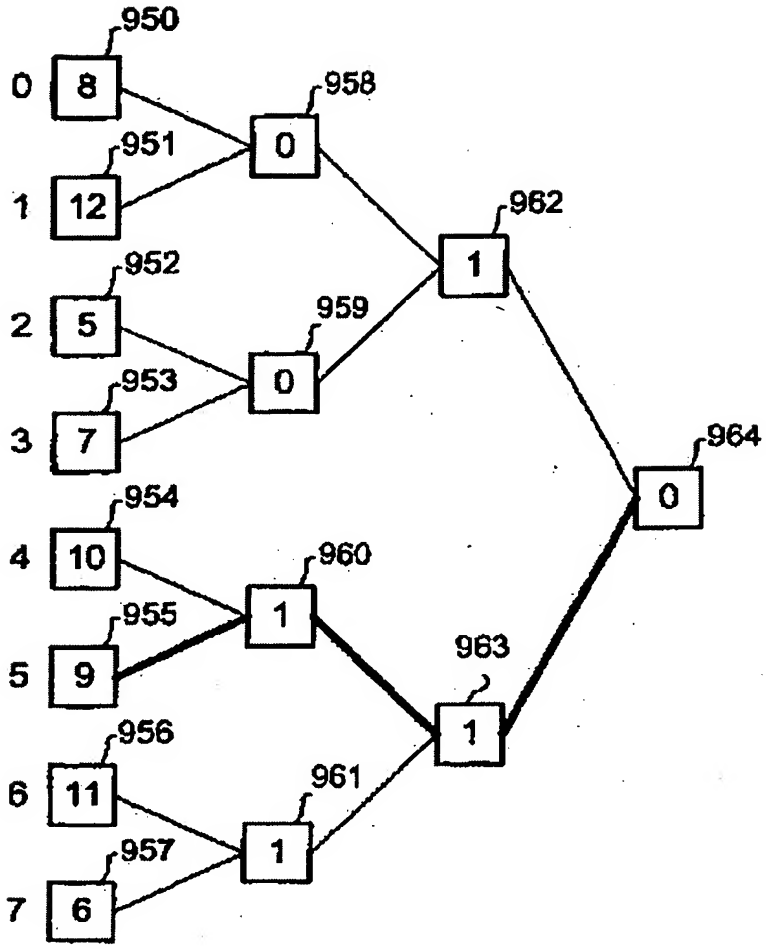
40-33

도면15


950
0
8
958
951
1
12
0
962
952
2
5
959
1
0
953
964
3
7
1
954
4
10
960
1
955
5
1
963
0
956
6
11
961
1
957
7
6

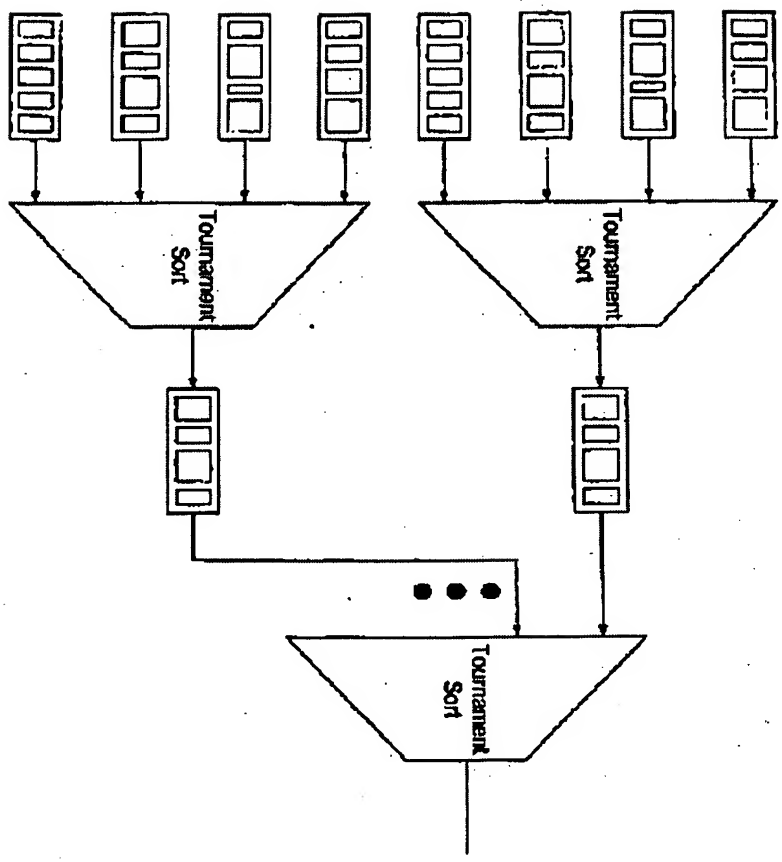
40-34

도면18


950
0
8
958
951
1
12
0
962
952
2
5
959
1
0
953
3
7
964
954
4
10
960
0
1
955
5
9
963
1
956
6
11
961
1
957
7
6

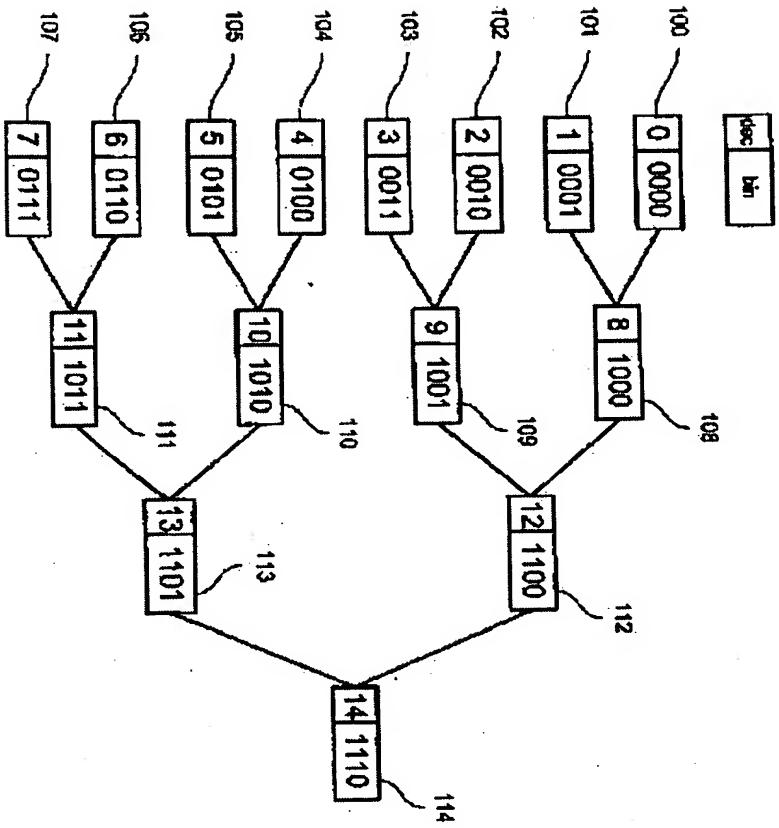
40-35

도면17



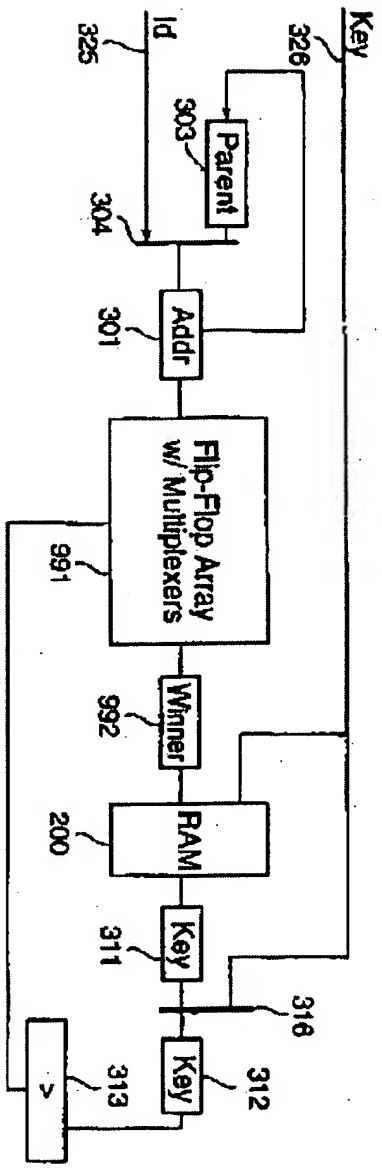
40-36

도면18


dec
bin
0 0000
1 0001
2 0010
3 0011
4 0100
5 0101
6 0110
7 0111
8 1000
9 1001
10 1010
11 1011
12 1100
13 1101
14 1110
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

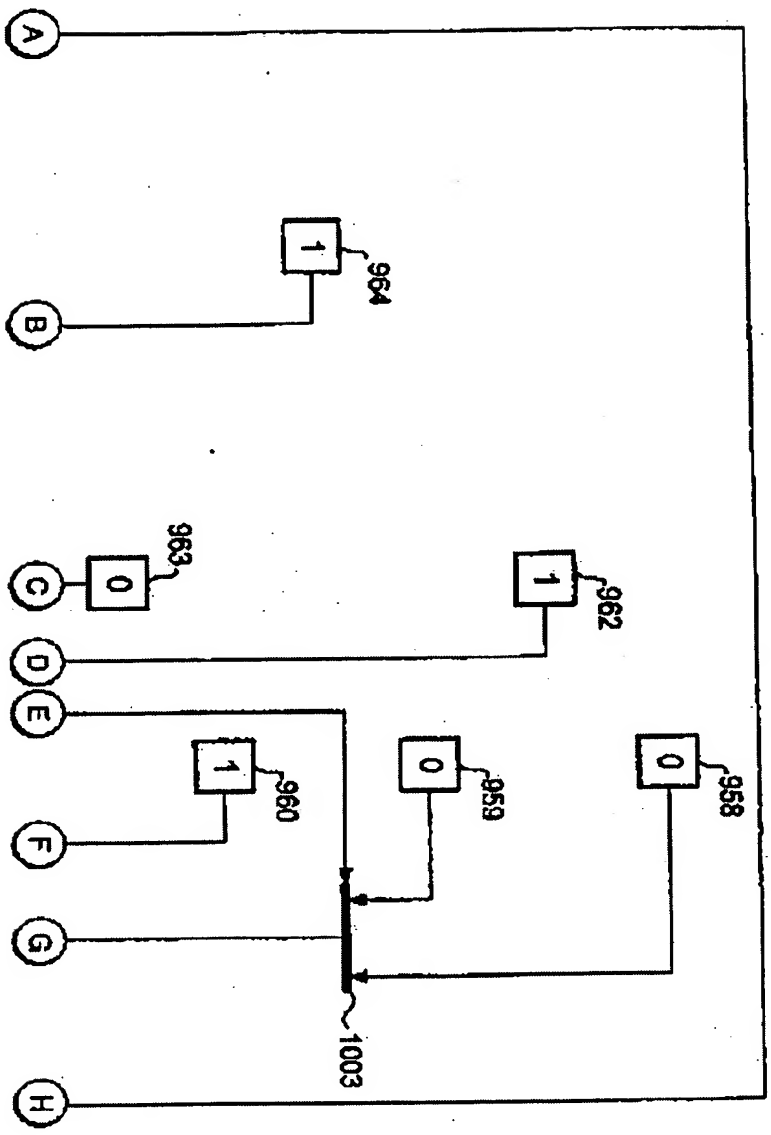
40-37

도면19


Key
326
Id
325
303
Parent
304
Addr
301
Flip-Flop Array
w/ Multiplexers
991
Winner
992
RAM
200
Key
311
316
Key
312
>
313

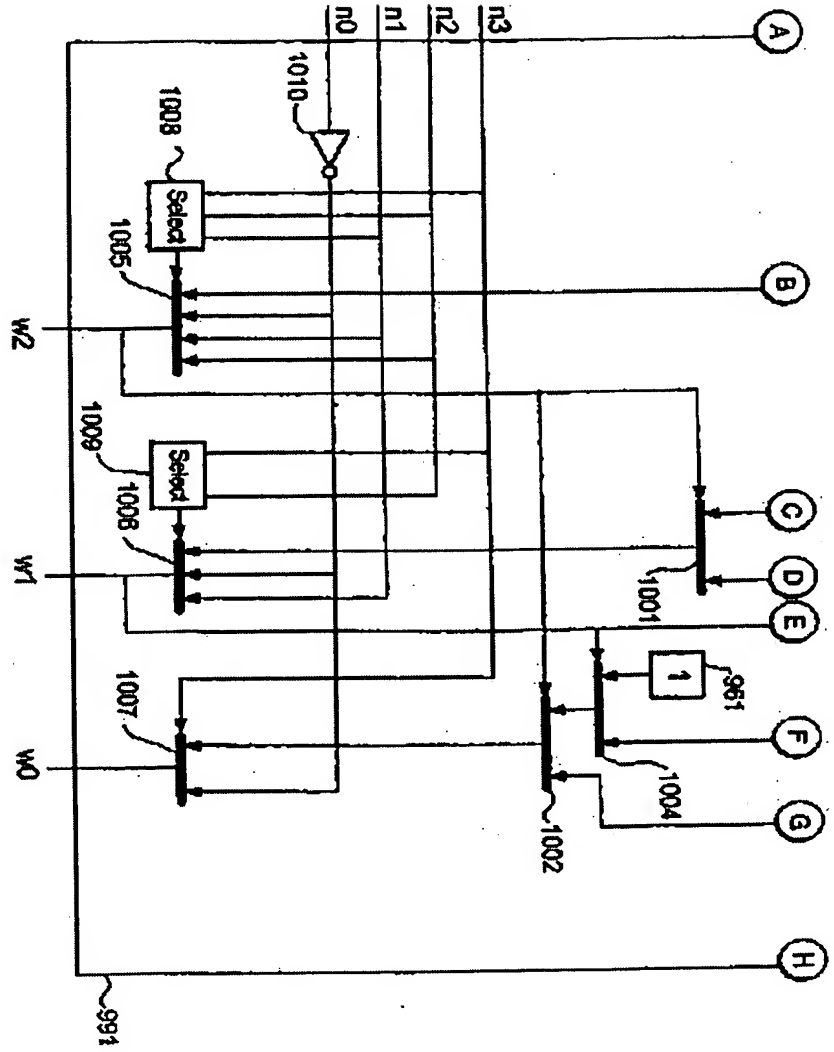
40-38

도면2d1



40-39

도면8


n0
n1
n2
n3
A
B
C
D
E
F
G
H
1010
1008
Select
1005
w2
1009
Select
1006
w1
1007
w0
1001
1
961
1004
1002
991

40-40

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.